TAMPEREEN TEKNILLINEN YLIOPISTO

JUUSE KOIVULA

MICROSOFT WORDIN LAAJENTAMINEN OHJELMISTOJEN MALLINNUSTYÖKALUKSI

Diplomityö

Tarkastajat: professori Kai Koskimies
yliassistentti Jari Peltonen
Tarkastajat ja aihe hyväksytty Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 04.04.2012

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
KOIVULA, JUUSE: Microsoft Wordin laajentaminen ohjelmistojen mallinnustyökaluksi
Diplomityö, 64 sivua, 3 liitesivua
Toukokuu 2012
Pääaine: Ohjelmistotuotanto
Tarkastajat: professori Kai Koskimies ja yliassistentti Jari Peltonen
Avainsanat: Mallinnustyökalu, laajennos, Microsoft Word, tietokanta, integrointi, dokumenttipohjainen

Tekstinkäsittelyohjelmistoja käytetään yleisesti analysoitaessa ja suunniteltaessa ohjelmistoja. Tällöin suunniteltavasta ohjelmistosta luodaan yleensä myös sitä kuvaava malli. Tekstinkäsittelyohjelmien avulla luodaan esimerkiksi dokumentteja, jotka kuvaavat suunniteltavan ohjelmiston mallia tai jotain sen osaa. Nykyiset mallinnustyökalut eivät kuitenkaan pysty hyödyntämään tekstinkäsittelyohjelmistoja kunnolla mallinnusprosessin osana, mistä johtuen tekstinkäsittelyohjelmilla luotavat dokumentit eivät ole osa mallia. Tämä johtaa muun muassa mallin ja dokumenttien välisiin eheysongelmiin.

Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksella kehitetään Trinity-mallinnus- ja työkaluympäristöä, jonka tarkoituksena on pyrkiä ratkaisemaan nykyisen mallinnustyön ongelmia. Ympäristön tarkoituksena on integroitua olemassa oleviin sovelluksiin, kuten toimisto-ohjelmiin, ja laajentaa niitä mallinnusominaisuuksilla.

Eräs ympäristöön integroitavista sovelluksista on Microsoft Word, johon toteutettua laajennosta tämä työ käsittelee. Laajennoksen tärkein tehtävä on mallidatan esittäminen ja muokkaaminen dokumenttimuotoisesti, mutta se tukee myös raporttien luomista sekä asiakirjojen muuntamista mallidataksi. Laajennoksen tarkoituksena on yhdistää mallinnustyökalujen mallinnusominaisuudet Wordiin siten, että mallidataa voidaan käsitellä Wordille ominaisilla työskentelytavoilla ja mekanismeilla. Lisäksi sille annettiin muita vaatimuksia, kuten mallinnustuen laajennettavuus ja tuki monelle samanaikaiselle käyttäjälle.

Työn tuloksena saatu Word-laajennos vastaa edellä mainittuihin vaatimuksiin ja tarjoaa hyvän pohjan jatkokehitykselle. Laajennoksen keskeisimmät haasteet ovat suorituskyky, joka ei ole vielä riittävällä tasolla, ja Wordin rajapinnan teknisten rajoitteiden kiertäminen. Tässä diplomityön kirjallisessa osassa esitellään Word-laajennoksen vaatimukset, sekä laajennoksen käyttö ja toteutus valituilta osin.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
KOIVULA, JUUSE: Extending Microsoft Word into a Document Based Software Modeling Tool
Master of Science Thesis, 64 pages, 3 Appendix pages
May 2012
Major: Software engineering
Examiners: Professor Kai Koskimies and assistant professor Jari Peltonen
Keywords: Modeling tool, extension, add-in, Microsoft Word, database, integration, document-based

Word processors are widely used in the analysis and design phases of software development. A model that describes the software under development is also usually created during the development process. Word processors are used to create documents that, for example, describe the model of the software that is being developed. However, current modeling tools do not take advantage of word processors as a part of the modeling process properly, and therefore, the created documents are not a part of the model. This leads, for instance, to consistency problems between the model and the documents.

Trinity is a modeling and software integration environment being developed at Department of Software Systems in Tampere University of Technology. Its goal is to find solutions to problems that currently exist in software modeling work. The environment integrates existing applications, such as office tools, and extends them with modeling support.

One of the applications integrated into Trinity is Microsoft Word. This was done with a Word extension that is the focus of this thesis. The main functionality of the extension is to present and edit model data in a document-based way, but it also supports report generation and creation of model data by using existing documents. The aim of the extension is to combine the modeling features of a modeling tool into Word in such a way that model data can be edited with Word's normal workflows and features. Other requirements for the extension are the extensibility of the modeling support and support for multiple simultaneous users.

The result of the work is a Word extension that fulfills all the aforementioned requirements. It also provides a good basis for future development. The main challenges of the extension are performance, which is currently not good enough, and working through the technical challenges posed by Word's API. This thesis presents the requirements for the extension, as well as its usage and implementation.

ALKUSANAT

Kiitän kaikkia Trinity-projektin jäseniä sekä Word-laajennoksen että siihen liittyvien asioiden suunnitteluun osallistumisesta. Kiitokset erityisesti myös työni tarkastajille Kai Koskimiehelle ja Jari Peltoselle työn ohjaamisesta ja arvokkaista kommenteista. Lämmin kiitos myös perheelleni ja ystävilleni kannustuksesta diplomityöni kirjoittamisessa.

Tampereella, 17. toukokuuta 2012

_____

Juuse Koivula

SISÄLTÖ

1. Johdanto 1
2. Taustakäsitteet ja -teknologiat 3
2.1 Ohjelmistojen mallintaminen 3
2.1.1 Mallinnuskielet ja metamallit 3
2.2 Microsoft Word 4
2.2.1 Wordin tyylimekanismi 4
2.2.2 Wordin oliomalli 5
2.2.3 Laajentaminen 6
2.3 Suunnittelu- ja arkkitehtuurimallit 6
2.3.1 Tarkkailija-suunnittelumalli 7
2.3.2 Kooste-suunnittelumalli 8
2.3.3 Tehdasmetodi- ja abstrakti tehdas -suunnittelumallit 8
2.3.4 Malli-näkymä-ohjain-arkkitehtuurimalli 9
2.3.5 Kerrosarkkitehtuuri 9
3. Tekstinkäsittelyohjelmistojen käyttö mallintamisessa 11
3.1 Tekstinkäsittelyohjelmistojen rooli mallinnusprosessissa 11
3.2 Nykyisen työkalutuen ongelmat ja puutteet 12
3.3 Mallinnustyökalun vaatimuksia 13
3.3.1 Mallinnuskäyttö 13
3.3.2 Asiakirjojen muuntaminen mallidataksi 14
3.3.3 Raportointi työkalun näkökulmasta 15
3.3.4 Ylläpitovaatimukset 15
3.4 Tekstinkäsittelyohjelmistoa tukevan mallinnusympäristön vaatimuksia . 15
3.4.1 Heterogeeninen mallinnusympäristö 16
3.4.2 Malli- ja näkymädata 16
3.4.3 Raportointi ympäristön näkökulmasta 16
3.4.4 Tiedon tallentaminen 18
4. Trinity-mallinnusympäristö 19
4.1 Trinityn yleiskuvaus 19
4.1.1 Trinity-ympäristön perusperiaatteet 19
4.1.2 Ympäristön rakenne 20
4.1.3 Malli- ja näkymädatan tallentaminen 20
4.1.4 Elementtipohjat 21
4.1.5 Suhteet ja näkymähierarkiat 22
4.1.6 Kaikille mallinnustyökaluille yhteiset toimintatavat ja mekanismit 22
4.2 Ympäristön tarjoamat palvelut 23
4.2.1 Hallintakäyttöliittymä 23

4.2.2 Informaatiopaneeli . . . 25
4.3 Trinity-ympäristön työkalulle asettamat vaatimukset . . . 26
4.3.1 Joustava mallintaminen . . . 26
4.3.2 Tietomalli ja käyttäjien samanaikainen työskentely . . . 26
4.3.3 Muut vaatimukset . . . 27
5. Wordin laajentaminen mallinnustyökaluksi . . . 29
5.1 Käyttöliittymä . . . 29
5.2 Näkymäelementit . . . 30
5.2.1 Näkymäelementtien rakenne . . . 30
5.2.2 Mallielementin ominaisuuksien esittäminen . . . 31
5.3 Elementtipohjat . . . 32
5.3.1 Elementtipohjien rakenne . . . 32
5.3.2 Metaominaisuudet . . . 33
5.3.3 Mallielementtien ominaisuudet . . . 34
5.3.4 Puhdas teksti ja elementtipohjien ulkonäkö . . . 34
5.4 Raporttipohjat . . . 35
5.5 Mallintaminen . . . 35
5.5.1 Näkymien avaaminen . . . 36
5.5.2 Näkymien sulkeminen . . . 36
5.5.3 Elementtien luominen . . . 37
5.5.4 Elementtien muokaaminen . . . 39
5.5.5 Elementtien poistaminen . . . 39
5.5.6 Sisällön kopiointi näkymästä . . . 40
5.5.7 Sisällön tuominen näkymään ulkopuolelta . . . 41
5.5.8 Värjäykset . . . 42
5.6 Asiakirjojen muuntaminen mallidataksi . . . 42
5.6.1 Muuntamisprosessin suorittaminen . . . 42
5.6.2 Muunnetun asiakirjan käsittely . . . 43
5.7 Raportointi . . . 43
5.8 Ylläpitotoiminnot . . . 44
5.8.1 Elementtipohjien hallinnointi . . . 44
5.8.2 Raporttipohjien hallinnointi . . . 45
5.9 Integraatio muihin työkaluihin . . . 45
6. Word-laajennoksen toteutus . . . 47
6.1 Suunnitteluperiaatteet . . . 47
6.2 Ympäristön arkkitehtuuri . . . 47
6.2.1 Tietokanta ja tietomalli . . . 47
6.2.2 Tietokantakomponentti . . . 49
6.2.3 Agenttiarkkitehtuuri . . . 49

6.3 Tool-ohjelmistokehys 50
6.4 Laajennoksen toteutustekniikat 51
6.5 Laajennoksen arkkitehtuuri 52
6.5.1 Model-kerros 52
6.5.2 ViewController-kerros 54
6.6 Word-laajennoksen erityisominaisuudet 56
6.6.1 Visio- ja Excel-näkymien näyttäminen Wordissa 56
6.6.2 Näkymähierarkiat 57
6.6.3 Raporttien luominen 58
6.6.4 Asiakirjojen muuttaminen mallidataksi 58
7. Työn arviointi 59
7.1 Vaatimusten toteutumisen arviointi 59
7.2 Trinity Word Add-inin tulevaisuus ja parannusehdotuksia 62
7.3 Aiheeseen liittyviä julkaisuja 62
8. Yhteenveto 64
Lähdeluettelo 65
Liite 1: Trinityn metametamalli 67
Liite 2: Tool-ohjelmistokehyksen ViewController-moduulin arkkitehtuuri 68
Liite 3: Word-laajennoksella luotu esimerkkikäyttötapaus 69

TERMIT JA NIIDEN MÄÄRITELMÄT

Annotointi	Metadatan lisääminen olemassaolevaan tietoon. (Engl. annotation)
Baseline-kopiointi	Eräs Trinity-ympäristön määrittelemistä mallien kopiointitavoista. Kopioituun malliin tehtyjä muutoksia voidaan verrata alkuperäiseen malliin.
BNF	Backus-Naur Form. Tapa ilmaista kontekstiriippumattomia kielioppeja.
COM	Component Object Model. Microsoftin määrittelemä rajapinta, jonka avulla ohjelmistokomponentit voivat kommunikoida ohjelmointikieliriippumattomasti.
DLL	Dynamic Link Library. Jaettujen kirjastojen toteutusteknologia Windows-ympäristössä.
Mallinnustyökalu	Ohjelmisto, jonka avulla mallinnetaan ohjelmistoja.
MOF	Meta-Object Facility. Malliohjautuvan ohjelmistokehityksen standardi, joka toimii UML:n metamallina.
.NET Framework	Microsoftin kehittämä ohjelmistokehys, joka on tarkoitettu COM:in korvaajaksi.
Oliomalli	Tässä työssä Microsoft Wordin oliomalli. Se tarjoaa joukon rajapintoja, joiden kautta voidaan käsitellä Word-prosessin tietoja.
Task pane	Microsoft Officen sisäinen ikkunointimekanismi.
Tekstinsyöttökursori	Wordissa tekstin seassa oleva kursori, joka kertoo mitä dokumentin kohtaa käyttäjä on muokkaamassa.
Trinity	Tampereen teknillisellä yliopiston Ohjelmistotekniikan laitoksella kehitetävä mallinnus- ja työkaluympäristö.
UML	Unified Modeling Language. Ohjelmistojen mallinnuksessa paljon käytetty mallinnuskieli.
URI	Uniform Resource Identifier. Merkkijono, joka identifioi jonkin abstraktin tai konkreettisen resurssin. [1]
Valintanauha	Microsoft Office 2007:ssa esitelty käyttöliittymäkomponentti, joka yhdistää valikot ja työkalurivit yhdeksi kokonaisuudeksi.
WYSIWYG	What You See Is What You Get. Tapa esittää tieto lähes sellaisena, miltä se näyttäisi lopullisessa tuotteessa.

1. JOHDANTO

Tekstinkäsittelyohjelmia käytetään yleisesti apuna ohjelmistojen analyysi- ja suunnitteluvaiheen aikana. Näiden vaiheiden aikana ohjelmistosta luodaan yleensä malli, joka kuvaa ohjelmiston rakennetta ja suunnitteluratkaisuja. Tällöin tekstinkäsittelyohjelmien avulla luodaan dokumentteja, joita voidaan käyttää esimerkiksi selittämään tarkemmin suunniteltavan ohjelmiston mallia tai jotain osaa. Dokumenteissa mallin sisältämä informaatio voidaan esittää monella tavalla, kuten tekstuaalisesti, taulukoina, kaavioina tai niiden yhdistelmänä. Tällainen esitystapa tarjoaa esimerkiksi puhtaasti kaaviomuotoisesta esitystavasta poikkeavan, ja sitä täydentävän näkökulman mallin sisältämän datan esittämiseen, luomiseen ja muokkaamiseen. Nykyiset mallinnustyökalut ja -ympäristöt eivät kuitenkaan pysty kunnolla hyödyntämään tekstinkäsittelyohjelmia osana mallinnusprosessia: luodut dokumentit eivät ole osa ohjelmiston mallia, vaikka ne esittävät sen sisältämää dataa. Tämä voi johtaa muun muassa eheysongelmiin dokumenttien ja mallin välillä.

Tähän ongelmaan on etsitty ratkaisua Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksella. Tutkimuksessa kehitetään Trinity-mallinnus- ja työkaluympäristöä, joka pyrkii ratkaisemaan mallinnustyössä esiintyviä ongelmia muun muassa integroitumalla olemassaoleviin ohjelmistoihin ja lisäämällä niihin mallinnusominaisuuksia. Ympäristössä päätettiin tutkia tekstinkäsittelyohjelmistojen ongelmallista suhdetta ohjelmistojen mallintamiseen integroimalla Microsoft Word osaksi ympäristöä. Tämä työ käsittelee Word-laajennoskomponenttia, joka laajentaa Wordin tekstinkäsittelyohjelmasta mallidataa ymmärtäväksi ohjelmistojen mallinnustyökaluksi. Laajennoksen tärkein toiminnallisuus on lisätä Wordiin tapa esittää, luoda ja muokata ympäristön sisältämää mallidataa. Tämän lisäksi laajennos tarjoaa mahdollisuuden raporttien luomiseen mallidatan pohjalta ja olemassaolevien asiakirjojen muuntamisen osaksi mallia.

Luvussa 2 esitellään työn kannalta oleelliset ohjelmistojen mallintamiseen liittyvät käsitteet, Microsoft Wordin toiminnot sekä suunnittelu- ja arkkitehtuurimallit. Luvussa 3 kerrotaan tekstinkäsittelyohjelmistojen käytöstä ohjelmistojen mallintamisessa ja esitellään sekä Word-laajennokselle että mallinnusympäristölle asetetut vaatimukset. Luvussa 4 esitellään Trinity-ympäristö ja sen työlle asettamat vaatimukset. Luku 5 esittelee työssä luodun Microsoft Word-laajennoksen. Sekä sen että mallinnusympäristön arkkitehtuuri kuvataan luvussa 6. Luvussa 7 arvioidaan to-

teutusta, esitetään parannusehdotuksia sekä esitellään aiheeseen liittyviä julkaisuja. Luku 8 sisältää työn yhteenvedon.

2. TAUSTAKÄSITTEET JA -TEKNOLOGIAT

Tässä luvussa kuvataan työn ymmärtämisen kannalta tärkeimmät taustatiedot ja teoria. Luvussa 2.1 esitellään ohjelmistojen mallintamiseen liittyviä käsitteitä, luvussa 2.2 esitellään Microsoft Wordin merkittäviä ominaisuuksia ja luvussa 2.3 kerrotaan työn kannalta tärkeistä suunnittelu- ja arkkitehtuurimalleista.

2.1 Ohjelmistojen mallintaminen

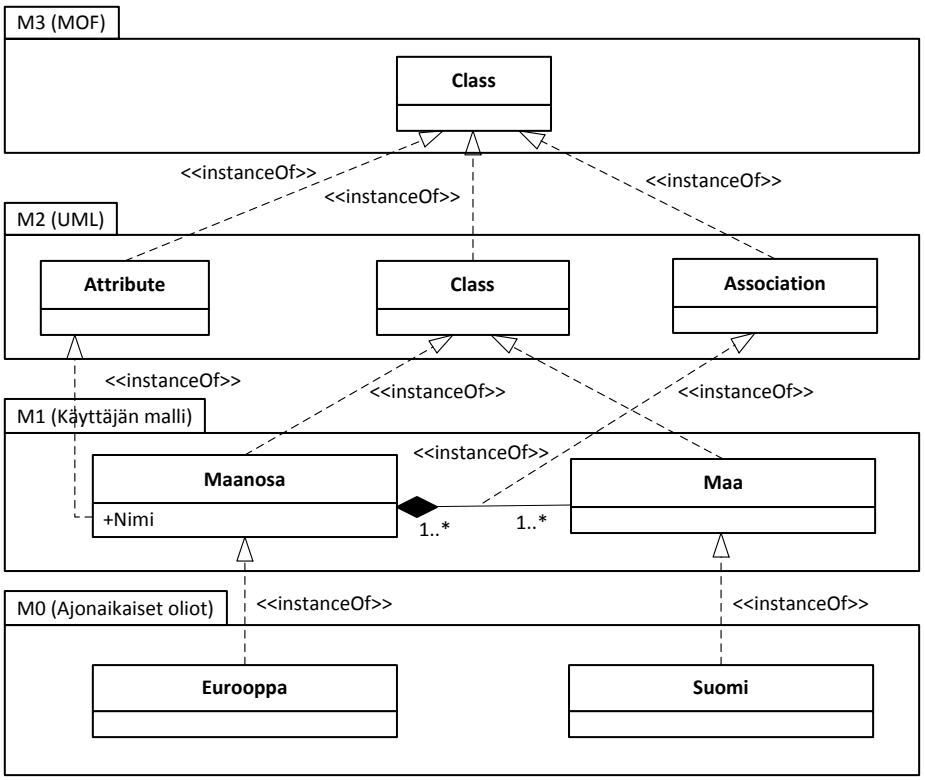
Ohjelmistoa suunniteltaessa siitä luodaan usein malli, joka kuvaa ohjelmistoa monesta eri näkökulmasta. Malli kuvaa ohjelmiston usein monilta erilaisilta abstraktiotasoilta, ja kullakin tasolla kuvataan ainoastaan sen kannalta merkitykselliset asiat. Mallien pääasiallinen käyttötarkoitus ohjelmistotyössä on ongelman analysointi, mutta niitä käytetään paljon myös esimerkiksi ihmisten välisen kommunikaation apuvälineenä ja dokumenttien laatimiseen. [8]

2.1.1 Mallinnuskielet ja metamallit

Mallinnuskielet ovat kieliä, jotka määrittelevät tapoja kuvata malleja. Ne esitetään yleensä graafisesti tai tekstuaalisesti. Tämän työn kannalta tärkein mallinnuskieli on Object Management Groupin (*OMG*) luoma ja ylläpitämä Unified Modeling Language (*UML*). Sen ensimmäinen versio julkaistiin vuonna 1997. Sen uusin versio kirjoitushetkellä on elokuussa 2011 julkaistu 2.4.1, joka sisältää 14 kaaviotyyppiä. Kaaviotyypit jaetaan kolmeen kategoriaan: rakenteellisiin kaaviotyyppeihin, käyttäytymiskaaviotyyppeihin ja vuorovaikutuskaaviotyyppeihin. [16]

Mallinnuskieli määrittelee mallissa käytettävät mallinnuskäsitteet. Näitä mallinnuskäsitteitä kutsutaan mallin *metamalliksi*. Myös metamalli voi perustua johonkin metamalliin, *metametamalliin*, joka voi edelleen perustua metamalliin, ja niin edelleen. Käytännössä kuitenkin ylin metataso on usein metametamalli, joka suunnitellaan siten, että se voi kuvata itse itsensä. Esimerkki tällaisesta metametamallista on Meta Object Facility (*MOF*), johon esimerkiksi UML perustuu. [8]

Kuvassa 2.1 esitetään UML:n nelitasoinen metamallihierarkia. Alin taso (M0) kuvaa ajonaikaisia olioita, joita mallinnettavat oliot kuvaavat. Toinen taso (M1), joka on samalla ensimmäinen metataso, kuvaa ohjelmiston varsinaista mallia. Kolmas taso (M2) kuvaa M1-tason tuottamiseen käytetyn mallinnuskielen. Ylimmällä tasolla (M3) kuvataan tapa, jolla mallinnuskielet on mallinnettu. Se myös kuvaa itse itsensä.



Kuva 2.1: UML:n mallitasot. Mukailtu lähteestä [15].

2.2 Microsoft Word

Microsoft Word on Microsoftin kehittämä tekstinkäsittelyohjelmisto. Se kuuluu Microsoft Office -tuoteperheeseen. Word tarjoaa WYSIWYG-pohjaisen tavan dokumenttien muokkaamiseen. Tässä luvussa kuvataan Wordin oliomalli, tyylimekanismi ja laajennusmekanismi.

2.2.1 Wordin tyylimekanismi

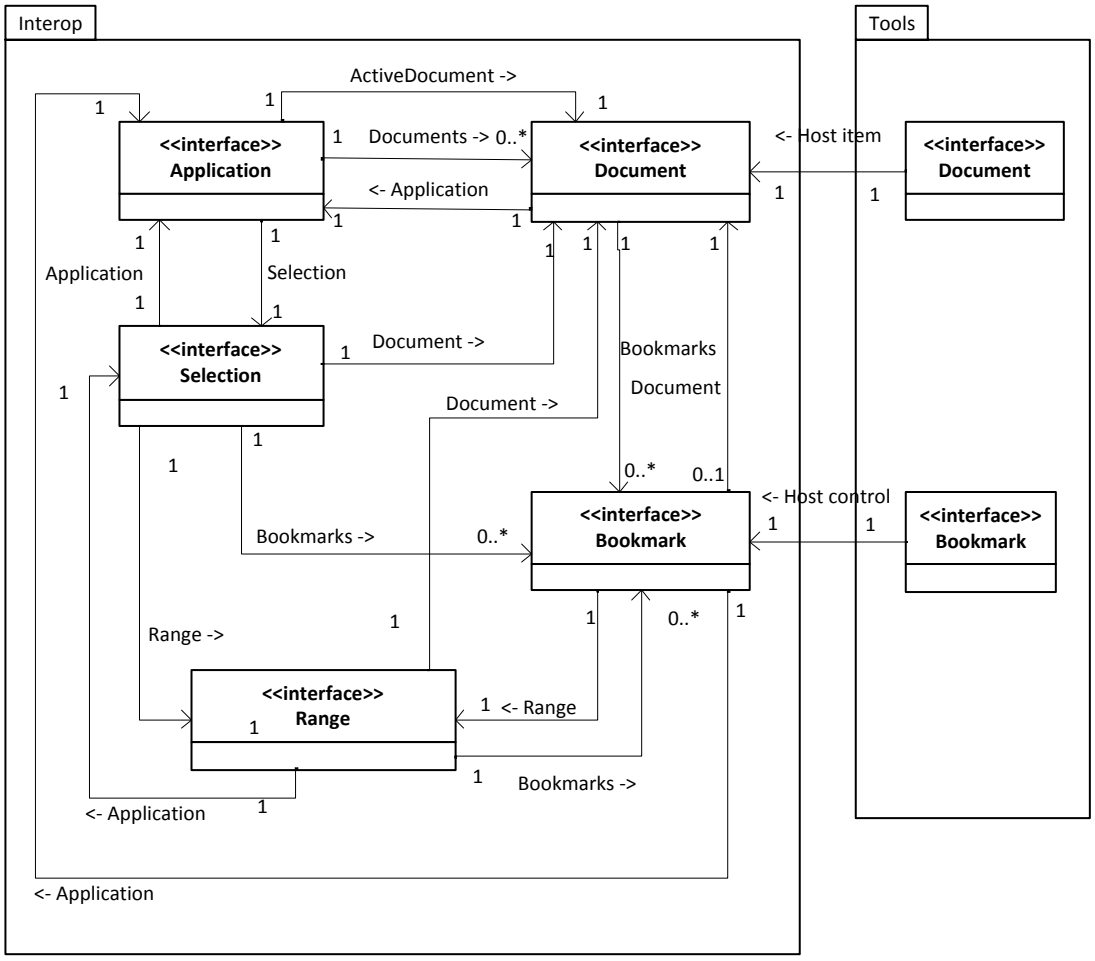
Word tarjoaa kaksi lähes päällekkäistä, mutta käyttötarkoituksiltaan erilaista työkalua dokumenttien kappalerakenteen ja ulkonäön muokkaamisen: *tyylimekanismin* sekä *tekstin ulkonäön ja kappalerakenteen muokkaamisen tekstiin käsin.* Tyylit ovat joko käyttäjän tai dokumenttipohjan määrittelemiä kokonaisuuksia, jotka vaikuttava tekstin ulkonäköön ja kappalerakenteeseen. Tyyli määrittelee tekstin ulkoasun ominaisuudet, kuten esimerkiksi käytetyn fontin nimen, fonttikoon, tekstin taustavärin ja onko teksti lihavoitu. Tämän lisäksi tyyliin liittyy informaatio siitä, kuinka laaja sen vaikutusalue on. Tämän työn kannalta tärkein vaikutusalue on yksittäinen merkki, mutta muita vaihtoehtoja ovat kokonainen kappale, lista ja taulukko. Tyylien tarkoitus on määritellä jonkin tekstialueen ulkonäkö korkealla tasolla.

Ulkonäön ja kappalerakenteen muokkaaminen käsin tarkoittaa valitun tekstinpätkän olemassaolevan tyylin yksittäisten ominaisuuksien ylikirjoittamista käsin. Sitä

ei ole tarkoitettu korvaamaan tyylimekanismia, vaan täydentämään sitä. Esimerkiksi tekstikappaleesta voidaan haluta nostaa esiin keskeisiä käsitteitä, jolloin niiden nimet voidaan lihavoida. Koska nämä tarpeet ovat yleensä erittäin väliaikaisia, kokonaan uusien tyylien tekeminen niitä varten olisi vaivalloista.

2.2.2 Wordin oliomalli

Word tarjoaa oliomallin, jonka kautta on mahdollista käsitellä sekä kyseisellä hetkellä auki olevaa Word-instanssia että siinä auki olevia dokumentteja ja niiden osia. Oliomallia voidaan käsitellä kaikilla .NET Frameworkia tukevilla ohjelmointikielillä, kuten C#:lla ja Visual Basic .NET:illä, sekä makrojen tekemiseen tarkoitetulla VBA-ohjelmointikielellä.



Kuva 2.2: Wordin oliomallin tärkeimmät rajapinnat. Mukailtu lähteestä [7].

Oliomalli tarjoaa käyttäjälle joukon rajapintoja, jotka jakautuvat kahteen nimiavaruuteen. *Interop-nimiavaruus* tarjoaa oliomallin perustoteutuksen. *Tools-nimiavaruus* laajentaa osaa Interop-nimiavaruuden rajapinnoista esimerkiksi lisäämällä niihin uusia tapahtumia (engl. event). Kaikki sen mukaiset oliot ovat alunperin Interop-

nimiavaruuden mukaisia olioita, jotka muunnetaan Tools-nimiavaruuden mukaisiksi. [11]

Oliomallin tämän työn kannalta tärkeimmät rajapinnat ovat *Application*, *Document*, *Bookmark*, *Range* ja *Selection*. Niiden väliset suhteet on esitetty kuvassa 2.2. Application kuvaa kyseisellä hetkellä auki olevaa Word-instanssia, Document yksittäistä auki olevaa dokumenttia, Bookmark dokumentissa olevaa kirjanmerkkiä ja Range jotain yhtäjaksoista tekstinpätkää. Selection kuvaa käyttäjän maalaamaa tekstialuetta, tai jos mitään tekstiä ei ole maalattu, tekstinsyöttökursorin sijaintia dokumentissa [10]. Näiden lisäksi oliomalli sisältää rajapintoja kuten yksittäistä tyyliä kuvaava *Style*, typografisia ominaisuuksia kuvaava *Font* ja tekstin seassa olevaa kuvaa kuvaava *InlineShape*.

2.2.3 Laajentaminen

Wordia on mahdollista laajentaa käyttäen sen tarjoamaa rajapintaa. Laajennokset voidaan jakaa kahteen pääryhmään: *add-in-laajennoksiin* ja *makrolaajennoksiin*. Nämä laajennostyypit eivät ole toisiaan poissulkevia, ja molemmat voivat olla käytössä yhtä aikaa. Erityyppiset laajennokset voivat kommunikoida keskenään [13], ja joidenkin toiminnallisuuksien toteuttaminen voi vaatia molempien käyttöä. Molemmilla laajennostyypeillä on pääsy Wordin oliomalliin.

Makrolaajennokset ovat Officen sisäinen mekanismi laajennosten tekemiseen. Niitä voidaan luoda Wordissa joko VBA-ohjelmointikielellä tai nauhoittamalla ohjelman käyttöä. Makrolaajennosten avulla on myös mahdollista korvata Wordin toimintoja kuten esimerkiksi kopioiminen, leikkaaminen ja liittäminen [22].

Add-in-laajennokset ovat .NET Frameworkin DLL-kirjastoja, jotka kommunikoivat Wordin kanssa COM-rajapinnan yli. Ne ajetaan samassa prosessissa Wordin kanssa. Niitä voidaan luoda millä tahansa .NET Frameworkia tukevalla ohjelmointikielellä. Add-in-laajennokset voidaan jakaa edelleen kahteen alikategoriaan: *asiakirjatason* ja *ohjelmatason* laajennoksiin. Tärkein ero näiden kahden tyypin välillä on se, että asiakirjatason laajennos on sidottu yksittäiseen asiakirjaan, kun taas ohjelmatason laajennos on sidottu Word-asennukseen. [12]

2.3 Suunnittelu- ja arkkitehtuurimallit

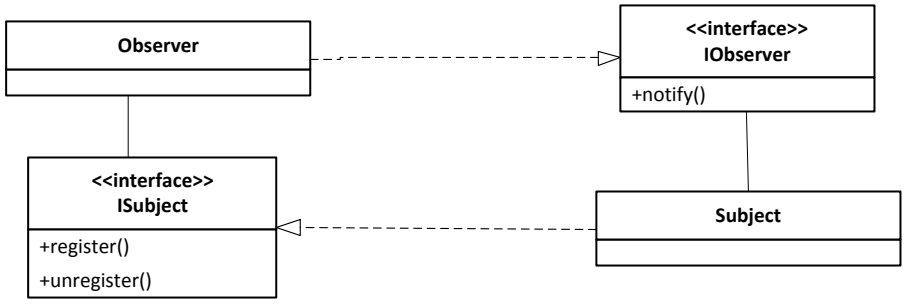
Suunnittelumallit ovat yleiskäyttöisiä ja yleisesti hyväksi todettuja suunnitteluratkaisuja, joita käytetään ratkaisemaan ohjelmistojen suunnittelussa vastaan tulevia ongelmia. Niitä käytetään yleisesti ohjelmistojen suunnittelussa. Ne toimivat myös kommunikaation apuvälineenä, sillä ne mahdollistavat suunnitteluratkaisuista keskustelemisen pelkästään niiden nimien avulla [9]. Suunnittelumallit alkoivat saavuttaa suosiota ohjelmistosuunnittelussa Erich Gamman, Richard Helmin, Ralph

Johnsonin sekä John Vlissidesin, eli niin kutsutun neljän koplan (englanniksi Gang of Four), vuonna 1994 julkaiseman *Design Patterns* -kirjan myötä.

Kirjassa suunnittelumallit jaetaan kolmeen ryhmään niiden käyttötarkoituksen mukaan: luomismalleihin, rakenteellisiin malleihin ja käytösmalleihin. Luomismallit kuvaavat erilaisia tapoja luoda olioita, rakenteelliset mallit olioiden ja luokkien välisiä suhteita ja niiden muodostamia rakenteita ja käytösmallit olioiden väliseen kommunikaatioon. Kirjassa myös esitellään 23 ohjelmistojen suunnittelumallia [3]. Tämänkin työn suunnittelussa ja toteutuksessa käytettiin hyväksi monia suunnittelumalleja, joista tärkeimmät on kuvattu tämän kappaleen aliluvuissa.

2.3.1 Tarkkailija-suunnittelumalli

Tarkkailija-suunnittelumalli (Observer pattern) on käytösmalli. Se kuvaa ohjelmistokomponenttien välistä kommunikaatiota tilanteessa, jossa monta ohjelmistokomponenttia on kiinnostunut yhdessä komponentissa tapahtuvista muutoksista. Se koostuu *tarkkailijasta* (observer) ja *kohteesta* (subject). Tarkkailija on kiinnostunut kohteessa tapahtuvista muutoksista ja rekisteröityy kohteelle sen tarkkailijaksi. Kohde ylläpitää listaa kaikista siitä kiinnostuneista tarkkailijoista, ja ilmoittaa nille siihen kohdistuneista muutoksista. Tarkkailijan ja kohteen välinen yhteys on löyhä, ja toteutetaan usein ohjelmointirajapintojen avulla. Kuvassa 2.3 esitetään suunnittelumallin yksi mahdollinen toteutus. [3]

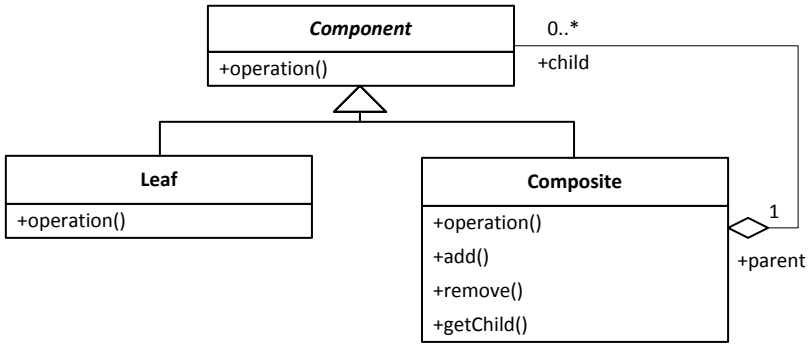


Kuva 2.3: Tarkkailija-suunnittelumalli. Mukailtu lähteestä [3].

.NET Framework mahdollistaa tarkkailija-suunnittelumallin toteuttamisen helposti *tapahtumien* (event) ja *delegaattien* (delegate) avulla. Kohteen ja tarkkailijan ei tarvitse määritellä rajapintoja, vaan tarkkailijat määrittelevät delegaatteja, jotka ne sitovat kohteen määrittelemiin tapahtumiin. Delegaatit ovat funktio-osoittimia, jotka sidotaan johonkin tapahtumaan, ja joiden tyyppi määräytyy tapahtuman tyypin perusteella. Kullekin delegaatille toteutetaan sitä vastaava funktio, joka suoritetaan, kun tapahtuma laukaistaan. [17]

2.3.2 Kooste-suunnittelumalli

Kooste-suunnittelumalli (Composite pattern) on rakenteellinen suunnittelumalli. Se määrittelee tavan puumaisten oliohierarkioiden kuvaamiseen. Sen toteutus esitetään kuvassa 2.4. Sekä Leaf- ja Composite-luokat periytyvät abstraktista Component-luokasta, joka kuvaa kaikille hierarkian solmuille yhteiset operaatiot. Leaf-luokka kuvaa sellaista hierarkian solmua, jolla ei voi olla lapsisolmuja, ja Composite-luokka sellaista, jolla voi olla lapsisolmuja. Composite-luokka sisältää kaikille solmuille yhteisten operaatioiden lisäksi operaatiot lasten lisäämiseen, hakemiseen ja poistamiseen. [3]

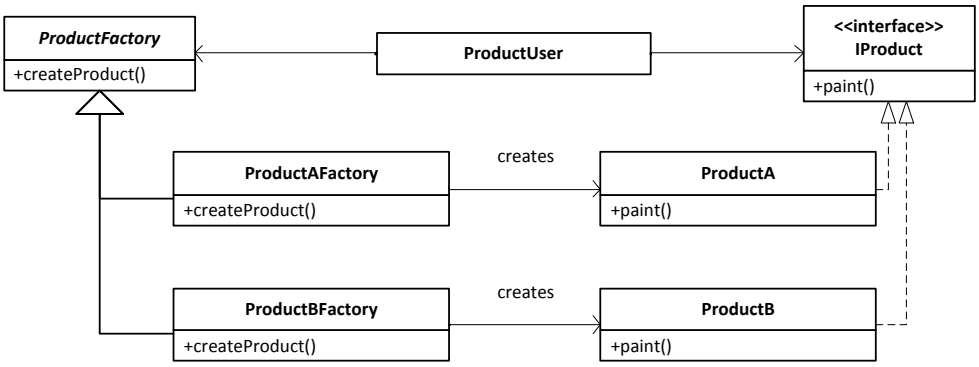


Kuva 2.4: Kooste-suunnittelumalli. Mukailtu lähteestä [3].

2.3.3 Tehdasmetodi- ja abstrakti tehdas -suunnittelumallit

Abstrakti tehdas -suunnittelumalli (Abstract factory pattern) on luomismalli. Se kapseloi yhteisen rajapinnan taakse joukon *tehdasluokkia*, joiden vastuulla on jonkin rajapinnan täyttävien olioiden luominen. Kuvassa 2.5 on esitetty yksi toteutus abstraktille tehtaalle. Kun jokin olio haluaa luoda uuden IProduct-rajapinnan toteuttavan olion, se kutsuu ProductFactory-luokan createProduct-funktiota. ProductUser ei saa tietää onko luotu olio tyyppiä ProductA vai ProductB eikä sitä, kumpaa ProductFactoryä sen luomiseen käytettiin. Abstrakteja tehtaita käytetään usein esitellyn tehdasmetodi-suunnittelumallin kanssa. [3]

Tehdasmetodi-suunnittelumalli (Factory method pattern) on luomismalli, jonka tarkoituksena on kätkeä jonkin rajapinnan toteuttavien olioiden luontilogiikka yhteen funktioon. Tehdasmetodina toimivan funktion sisäinen logiikka päättelee minkä tyyppinen olio luodaan ja palauttaa paluuarvona uuden rajapinnan luotuun olioon, joten kutsuja ei saa tietää luodun olion tarkkaa tyyppiä. Tehdasmetodeita voidaan käyttää myös tilanteissa, joissa luokan rakentajaa ei voida kuormittaa esimerkiksi parametrien tyyppien takia. Kuvassa 2.5 on esimerkki suunnittelumallin toteutuksesta: Kaikki IProduct-rajapinnan toteuttavat oliot luodaan ProductFactory-luokan createProduct-funktion avulla. [3]

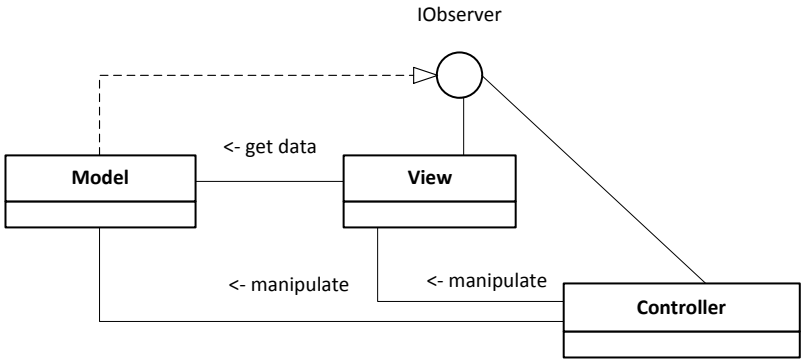


Kuva 2.5: Abstrakti tehdas -suunnittelumalli. Mukailtu lähteestä [3].

2.3.4 Malli-näkymä-ohjain-arkkitehtuurimalli

Malli-näkymä-ohjain (MVC, model-view-controller) on arkkitehtuurimalli, jonka perusperiaatteena on erottaa sovelluksen sisältämä data, sen esitystapa ja sovelluslogiikka omiin komponentteihinsa [9]. Tämä parantaa muun muassa käyttöliittymän muokattavuutta.

MVC-mallista on olemassa monia erilaisia versioita. Tässä esitelty versio toimii seuraavalla tavalla: Sovelluksen tietomalli ja sen muokkaamiseen käytetyt operaatiot esitetään *mallikomponentissa*, ja sovelluksen käyttöliittymälogiikka sijaitsee *näkymäkomponentissa*. Näiden välillä toimii *ohjainkomponentti*, joka päivittää käyttöliittymän kautta tehdyt muutokset malliin ja toisin päin. Malli- ja näkymäkomponentit kertovat muutoksistaan ohjaimelle luvussa 2.3.1 kuvatun tarkkailijasuunnittelumallin avulla. Tämä MVC-mallin variaatio on esitetty kuvassa 2.6. Yleisenä MVC-mallin kanssa sovellettavana käytäntönä on, että jokaiselle näkymäkomponentin luokalle luodaan siitä vastaava ohjainluokka.



Kuva 2.6: Malli-näkymä-ohjain-arkkitehtuurimalli. Mukailtu lähteestä [9].

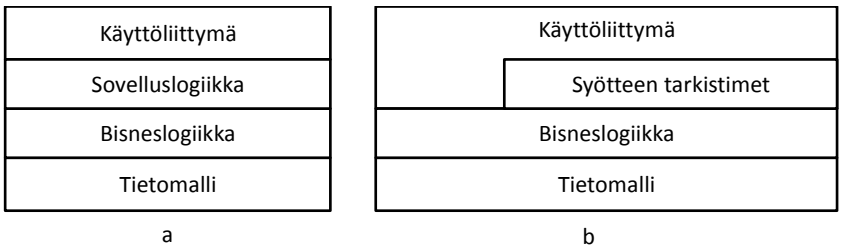
2.3.5 Kerrosarkkitehtuuri

Kerrosarkkitehtuuri on arkkitehtuurityyli, jossa ohjelmisto jaetaan kerroksiin jonkin abstrahointiperiaatteen perusteella. Kerrokset järjestetään päällekkäin abstraktiota-

son mukaan nousevaan järjestykseen, ja ylempien kerrosten komponentit toteutetaan alempien kerrosten tarjoamien palveluiden avulla. [9]

Eräs puhtaan kerrosarkkitehtuurin periaatteista on se, että funktiokutsut kulkevat ainoastaan ylemmistä kerroksista alempiin. Tällä saavutetaan se, että alemmat kerrokset eivät riipu niitä ylempänä olevasta toteutuksesta. Käytännössä tämä on kuitenkin harvinaista, ja alempien kerrosten palvelut tarvitsevat usein myös niiden yläpuolella olevien kerrosten palveluita. Tämä voidaan kuitenkin toteuttaa siten, että tästä syntyvä riippuvuus on mahdollisimman löyhä, esimerkiksi takaisinkutsujen avulla. [9]

Puhtaassa kerrosarkkitehtuurissa funktiokutsut eivät saisi myöskään ohittaa kerroksia. Esimerkki tällaisesta kerrosarkkitehtuurista on kuvattu kuvassa 2.7 a. Myös kerrosten ohitukset ovat yleisiä, sillä niiden avulla voidaan esimerkiksi parantaa ohjelmiston tehokkuutta. Kuvassa 2.7 b on esitetty esimerkki kerrosarkkitehtuurista, jossa on ohituksia. [9]



Kuva 2.7: a: Kerrosarkkitehtuuri, jossa ei ole kerrosten välisiä ohituksia. b: Kerrosten välisiä ohituksia sisältävä kerrosarkkitehtuuri. Mukailtu lähteestä [9].

3. TEKSTINKÄSITTELYOHJELMISTOJEN KÄYTTÖ MALLINTAMISESSA

Luvuissa 3.1 ja 3.2 kerrotaan asiakirjojen rooleista ohjelmistojen mallinnusprosessissa ja nykyisen työkalutuen ongelmista ja puutteista. Luvussa 3.3 esitellään tekstinkäsittelyohjelmistopohjaisen mallinnustyökalun yleisiä vaatimuksia ja 3.4 esitellään tällaisen työkalun mallinnusympäristölle esittämiä vaatimuksia, joihin vastaamalla nämä ongelmat voidaan ratkaistaan.

3.1 Tekstinkäsittelyohjelmistojen rooli mallinnusprosessissa

Tekstinkäsittelyohjelmia käytetään paljon ohjelmiston analyysi- ja suunnitteluvaiheiden aikana esimerkiksi muodostamalla dokumentteja, jotka mahdollistavat ohjelmistosta luotavan mallin informaation esittämisen. Dokumenttien avulla mallista pystytään tarkastelemaan ominaisuuksia, jotka eivät tulisi esiin esimerkiksi kaaviomuotoisen esitystavan kautta. Esimerkiksi ohjelmiston arkkitehtuurin suunnitteluratkaisut voidaan esittää luokkakaavioina, mutta perustelut niiden valintaan kuvataan tekstimuotoisesti.

Ohjelmiston analyysi- ja suunnitteluvaiheessa muodostettavat dokumentit ovat pääasiassa asiakirjamuotoisia ja tekstinkäsittelyohjelmistojen ensisijainen käyttötarkoitus on asiakirjojen luominen. Tästä johtuen tekstinkäsittelyohjelmistot ovat pääasiallisia työkaluja näiden dokumenttien luomiseen. Esimerkkejä mallinnusprosessissa luotavista ja käytettävistä dokumenteista ovat mallinnettavan järjestelmän dokumentaatio, mallin perusteella tehdyt tekstimuotoiset raportit ja mallinnettavaan ohjelmistoon liittyvät tekniset standardit.

Ohjelmistojen suunnittelussa yleisesti käytettyjä työskentelytapoja ovat dokumenttipohjainen ja mallipohjainen työskentelytapa. Dokumenttipohjaisessa työskentelyssä järjestelmän elinkaaren analyysi- ja suunnitteluvaiheessa pääpaino on siitä luoduissa dokumenteissa, ja suunniteltava järjestelmä toteutetaan niiden pohjalta. Tälle vastapainona on mallipohjainen työskentely, jossa tärkein lopputuote on ohjelmistosta luotava malli. Tällöin dokumentit ovat analyysi- ja suunnitteluvaiheen sivutuote, ja järjestelmä toteutetaan mallin perusteella. Molemmissa työskentelytavoissa on sekä hyvät että huonot puolensa, ja ideaalisesti pitäisi pyrkiä yhdistämään niiden hyvät puolet.

3.2 Nykyisen työkalutuen ongelmat ja puutteet

Nykyiset mallinnusympäristöt eivät tue kunnolla tekstinkäsittelyohjelmistojen käyttöä mallinnustyökaluina. Niiden avulla luotavat dokumentit ovat informaaleja, eivätkä ne ole osa ohjelmiston varsinaista mallia. Dokumenttien sisältämä informaatio liittyy kuitenkin kiinteästi malliin, ja sitä voidaan ajatella mallidatana samoin kuin esimerkiksi luokkakaavion luokkia.

Dokumenttien ja muun mallidatan välille **ei synny minkäänlaista takaisinkytkentää, ja dokumentit täytyy päivittää käsin** aina kun mallidata mallissa muuttuu ja toisin päin. Mallidatan päivittäminen käsin molempiin mallin osiin vie aikaa varsinaiselta analyysi- ja suunnittelutyöltä. Se myös altistaa inhimillisille virheille, koska ohjelmistojen mallintamisen iteratiivisen luonteen vuoksi malli muuttuu usein. Suurin ongelma on kuitenkin **tiedon eheyden vaarantuminen**, sillä malliin tehdyt muutokset voidaan joko unohtaa päivittää tai jätetään tarkoituksella päivittämättä dokumentteihin tai toisin päin. Jos dokumenttien ja mallin välillä on ristiriita, kumpi on oikeassa?

Tämä tulee ongelmaksi sekä **muodostettaessa dokumentteja olemassaolevan mallidatan pohjalta** että **luotaessa uutta mallidataa** tekstinkäsittelyohjelmiston avulla. Esimerkki dokumenttien luomisesta olemassaolevan mallidatan perusteella on mallinnettavan ohjelmiston dokumentaation muodostaminen. Dokumentaatiossa ohjelmiston arkkitehtuuri kuvataan yleensä joukkona kaavioita, mutta siitä muodostetaan myös tekstuaalinen kuvaus. Se saadaan kuvattua tällä tavalla tarkemmin ja luonnollisemmin kuin pelkillä kaavioilla. Sanallinen esitysmuoto on luonnollinen esitystapa esimerkiksi perusteluille valittujen suunnittelu- ja arkkitehtuuriratkaisujen takana. Koska nykyiset mallinnusympäristöt eivät tue mallidatan syöttämistä tällä tavalla, ohjelmiston dokumentaatio jää informaaliksi.

Esimerkki uuden mallidatan luomisesta tekstinkäsittelyohjelman avulla on käyttötapausten kuvausten luominen. Niissä selitetään jonkin käyttötapauksen läpiviemiseen tarvittavat askeleet ja siinä osana olevien toimijoiden vastuut niiden suorittamisen aikana. Niille luonnollinen esitysmuoto on sanallinen esitys, ja siten niitä tehdään käyttäen tekstinkäsittelyohjelmistoja. Jos käyttötapausten kuvaukset halutaan muuttaa osaksi ohjelmiston formaalia mallia nykyisillä mallinnustyökaluilla, ne täytyy syöttää ympäristöön sille ei-ominaisessa muodossa, kuten esimerkiksi kaaviona.

Tämä sama pätee ulkoisissa, **asiakirjamuotoisissa lähteissä sijaitsevan tiedon hyödyntämiseen mallinnuksessa**. Malleissa on usein tarve viitata esimerkiksi teknisiin standardeihin, joiden esitysmuoto on yleensä asiakirja. Mallinnettavalle ohjelmistolle on esimerkiksi saatettu esittää vaatimus, että se joutuu täyttämään jotain teknisiä standardeja. Tällöin malleista halutaan viitata näihin standardeihin

merkitsemällä, että jokin ohjelmiston suunnitteluratkaisu on jonkin standardin, tai sen osan, mukainen. Nykyisillä mallinnustyökaluilla ja -ympäristöillä ei ole kuitenkaan mahdollista analysoida valmiita asiakirjoja ja tuottaa ympäristöön mallidataa niiden pohjalta.

Kehittyneimmät nykyiset mallinnustyökalut mahdollistavat **automaattisten raporttien luomisen** mallidatan pohjalta [20][18]. Nekin kuitenkin kärsivät osittain edellämainituista ongelmista. Luodut raportit ovat tässäkin tapauksessa staattisia asiakirjoja, joiden kautta mallidatan muokkaaminen mallissa on mahdotonta. Tämän lisäksi vaikka raporttiasiakirjat luodaankin automaattisesti, uusien raporttien versioiden luominen on edelleen käyttäjien vastuulla.

Mainittujen ongelmien perusteella voidaan todeta, että nykyisten mallinnustyökalujen asiakirjatuessa on selkeitä puutteita. Mallinnusprosessissa joudutaan tekemään mallinnustyön näkökulmasta turhia työvaiheita, kuten kopioimaan varsinaiseen malliin tehtyjä muutoksia ohjelmiston dokumentaatioon, joiden automatisoimattomuus johtaa hukattuun aikaan ja inhimillisiin virheisiin.

Yksi vaihtoehto ongelmien ratkaisemiseksi on dokumenttien sisältämän informaation ottaminen osaksi mallia. Tämä tarkoittaa, että on toteutettava mallinnustyökalu, joka yhdistää ohjelmistojen mallinnustyökalut ja tekstinkäsittelyohjelmistot. Tällainen työkalu tehostaisi mallinnusprosessia tarjoamalla luonnollisen tavan esittää ja luoda tekstuaalista mallidataa siihen soveltuvilla työkaluilla, tehden mallinnusprosessista joustavamman. Tällä on myös suora yhteys ohjelmiston laatuun, koska esimerkiksi se, että kuinka hyvin tuotettu dokumentaatio kuvaa lopullista ohjelmistoa, on suoraan verrannollinen ohjelmiston ylläpidettävyyteen [4, s. 56].

3.3 Mallinnustyökalun vaatimuksia

Luvussa 3.2 mainitut ongelmat muodostavat pohjan asiakirjapohjaisen mallinnustyökalun vaatimuksille. Niille on annettu nimi ja tunniste. Tunnisteen "M"-etuliite tulee sanasta "mallinnustyökalu". Vaatimukset voidaan jakaa mallinnuskäytön, asiakirjojen mallidataksi muuntamisen, raportointimekanismin toteuttamisen ja ympäristön ylläpidon esittämiin vaatimuksiin.

3.3.1 Mallinnuskäyttö

Työkalulla mallintamisen pitää olla järkevää tekstinkäsittelyohjelmistoihin liittyvät yleiset työskentelytavat huomioon ottaen. Esimerkiksi olemassaolevia malli- ja näkymäelementtejä tulee pystyä muokkaamaan tekstinkäsittelyohjelmistoille tyypillisillä muokkausmekanismeilla, kuten syöttämällä vapaamuotoista sisältöä näppäimistöltä sekä muokkaamalla asiakirjan tyylejä. Tämä asettaa sille joukon vaatimuksia, jotka on kuvattu taulukossa 3.1.

Taulukko 3.1: Mallinnustyökalun käyttöliittymän vaatimukset

Tunniste ja nimi	Kuvaus
M01, Työskentelytapa	Työkalulla mallintamisen on noudatettava tekstinkäsittelyohjelmistojen yleisiä työskentelytapoja.
M02, Elementtien muokkaaminen	Olemassaolevia malli- ja näkymäelementtejä on pystyttävä muokkaamaan tekstinkäsittelyohjelmistojen tiedonmuokkausmekanismeilla.
M03, Elementtien luominen pohjien avulla	Uusia mallielementtejä on pystyttävä luomaan ennalta määriteltyjen mallielementtipohjien avulla.
M04, Elementtien luominen vapaamuotoisesti	Uusia mallielementtejä on pystyttävä luomaan tuottamalla näkymään vapaamuotoista sisältöä ja merkitsemällä mallielementtien sijainnit jälkikäteen.
M05, Mallielementtien ulkopuolinen näkymädata	Näkymien sisältämä, mutta mihinkään mallielementtiin kuulumaton sisältö on tallennettava näkymäelementteihin, jotka eivät liity mihinkään mallielementteihin.

3.3.2 Asiakirjojen muuntaminen mallidataksi

Olemassaolevien, staattisten asiakirjojen muuntaminen mallidataksi ympäristöön on tärkeä toiminnallisuus tekstinkäsittelyohjelmistopohjaisessa mallinnustyökalussa, sillä se mahdollistaa mallidatan tuomisen ympäristöön esimerkiksi teknisistä standardeista. Jotta muunnettuihin asiakirjan osiin pystyttäisiin viittaamaan muualta ympäristöstä, asiakirja on jaettava mallielementteihin. Muunnosprosessin on myös kuormitettava käyttäjää mahdollisimman vähän, joten se täytyy olla mahdollista viedä läpi mahdollisimman vähäisellä kontaktilla käyttäjään. Muunnostoiminnallisuuden työkalulle asettamat vaatimukset on kuvattu taulukossa 3.2.

Taulukko 3.2: Asiakirjojen mallidataksi muuntamisen vaatimukset

Tunniste ja nimi	Kuvaus
M06, Asiakirjojen muuntaminen mallidataksi	Työkalun avulla on pystyttävä muuntamaan kokonaisia asiakirjoja mallidataksi ympäristöön.
M07, Muunnettujen asiakirjojen käsitteleminen	Muunnettuja asiakirjoja on pystyttävä käsittelemään samalla tavalla kuin mallinnusprosessin ulkopuolisia asiakirjoja.
M08, Asiakirjojen analysointi	Muunnosprosessissa asiakirjan rakenne on analysoitava ja pilkottava mallielementtihierarkiaksi, joista jokainen elementti kuvaa yhtä asiakirjan lukua tai alilukua.
M09, Muunnosprosessin automatisointi	Muunnosprosessi on pystyttävä suorittamaan automaattisesti kokonaiselle asiakirjalle.

3.3.3 Raportointi työkalun näkökulmasta

Yksi tekstinkäsittelyohjelmistojen käyttökohteista ohjelmistojen mallintamisessa on dokumenttimuotoisten raporttien luominen mallidatan pohjalta, ja työkalun on tuettava tätä. Raporttien luonti pitää myös pystyä käynnistämään automaattisesti esimerkiksi web-käyttöliittymän kautta. Raportointimekanismin työkalulle asettamat vaatimukset on kuvattu taulukossa 3.3.

Taulukko 3.3: Raportoinnin työkalulle asettamat vaatimukset

Tunniste ja nimi	**Kuvaus**
M10, Raporttien luominen	Työkalulla on pystyttävä luomaan asiakirjamuotoisia raportteja mallidatan perusteella.
M11, Raporttien tallentaminen	Luodut raportit on tallennettava ympäristön tietovarastoon mallielementteinä.
M12, Raporttien yhdistäminen elementteihin	Raportteja on pystyttävä yhdistämään malleihin ja toisiin mallielementteihin.
M13, Raporttien luonnin etäkäynnistäminen	Raportin luominen on pystyttävä käynnistämään ympäristön muista työkaluista, kuten www-käyttöliittymästä.
M14, Raporttien luominen palvelimella	Raportin luominen on pystyttävä viemään läpi palvelimella ilman suoraa kontaktia käyttäjään.

3.3.4 Ylläpitovaatimukset

Koska heterogeenisen mallinnusympäristön mallinnustyökalut voivat erota toisistaan merkittävästi, jokaisen mallinnustyökalun on huolehdittava itse omiin elementti- ja raporttipohjiinsa liittyvistä ylläpitotoiminnoista. Niiden esittämät vaatimukset on kuvattu taulukossa 3.4.

Taulukko 3.4: Muut vaatimukset

Tunniste ja nimi	**Kuvaus**
M15, Elementtipohjien ylläpito	Työkalun avulla on pystyttävä luomaan uusia ja muokkaamaan olemassaolevia elementtipohjia helposti.
M16, Raporttipohjien ylläpito	Työkalun avulla on pystyttävä luomaan uusia ja muokkaamaan olemassaolevia raporttipohjia helposti.

3.4 Tekstinkäsittelyohjelmistoa tukevan mallinnusympäristön vaatimuksia

Tekstinkäsittelyohjelman integroiminen mallinnusympäristöön asettaa vaatimuksia myös itse mallinnusympäristölle. Vaatimusten tarkoituksena on, että vastaamalla

niihin voidaan muodostaa mallinnusympäristö, joka tukee mallidatan esittämistä ja muokkaamista tekstinkäsittelyohjelmalla muiden esitystapojen lomassa. Vaatimukset voidaan jakaa seuraaviin alikategorioihin: heterogeenisen mallinnusympäristön vaatimukset, näkymädatan vaatimukset, raportoinnin asettamat vaatimukset ja tiedon tallentamisen vaatimukset. Tunnisteen "Y"-etuliite tulee sanasta "ympäristö".

3.4.1 Heterogeeninen mallinnusympäristö

Monenlaisia mallidatan esitysmuotoja tukevan ympäristön on oltava tukemiensa mallinnustyökalujen suhteen heterogeeninen, ja sen on pystyttävä integroimaan uusia mallinnustyökaluja. Tämä asettaa sille vaatimuksia, jotka on kuvattu taulukossa 3.5.

Taulukko 3.5: Heterogeenisen mallinnusympäristön vaatimukset

Tunniste ja nimi	**Kuvaus**
Y01, Heterogeenisuus	Ympäristön on tuettava monia erilaisia mallinnustyökaluja.
Y02, Laajennettavuus	Ympäristöön on pystyttävä lisäämään uusia mallinnustyökaluja.
Y03, Mallidatan esittäminen	Mallidataa on pystyttävä esittämään monenlaisilla mallinnustyökaluilla.
Y04, Mallidatan luominen ja muokkaaminen	Mallidataa on pystyttävä luomaan ja muokkaamaan monenlaisilla mallinnustyökaluilla.

3.4.2 Malli- ja näkymädata

Malleihin tulee pystyä tekemään monentyyppisiä näkymiä, jotka voidaan avata siinä työkalussa, johon kyseinen näkymätyyppi liittyy. Ympäristön tavalle esittää malli- ja näkymädata esitettyjen vaatimusten tarkoituksena on varmistaa, että mallidata pystytään esittämään monella esitystavalla. Nämä vaatimukset on listattu taulukossa 3.6.

3.4.3 Raportointi ympäristön näkökulmasta

Jotta koko ympäristön laajuinen, helposti laajennettava raportointimekanismi voidaan toteuttaa, myös mallinnusympäristölle on asetettava vaatimuksia. Koska raportointi voidaan käynnistää jostain ympäristön muusta työkalusta, kuten web-käyttöliittymästä, luodut raportit on pystyttävä tallentamaan keskitetysti. Taulukossa 3.7 on listattu nämä vaatimukset. Ne liittyvät luvussa 3.3.3 esiteltyihin raportointia tukevan mallinnustyökalun vaatimuksiin.

Taulukko 3.6: Malli- ja näkymädatan vaatimukset

Tunniste ja nimi	**Kuvaus**
Y05, Näkymädatan erottaminen mallidatasta	Mallidata ja sen graafinen esitystapa, näkymädata, täytyy erottaa toisistaan.
Y06, Näkymädatan tallentaminen	Näkymädatan tallentamistapa pitää olla sellainen, että sen avulla voidaan tallentaa kaikkien ympäristön työkalujen tarvitsema työkalukohtainen näkymädata.
Y07, Monenlaiset näkymät	Malleihin on pystyttävä luomaan monentyyppisiä näkymiä, jotka voidaan avata siinä työkalussa, johon kyseinen näkymätyyppi liittyy.
Y08, Näkymien kytkeminen toisiinsa	Näkymiä on pystyttävä kytkemään toisiinsa asiakirjojen kappalejakoa muistuttaviksi näkymähierarkioiksi, joissa voi olla osana monentyyppisiä näkymiä.
Y09, Näkymätyyppien lisääminen	Ympäristöön on pystyttävä lisäämään helposti uusia malli- ja näkymätyyppejä menettämättä aiemmin luotua mallidataa.
Y10, Mallielementtiviittaukset	Ympäristön on sallittava mallielementtien viittaaminen toisiinsa sekä mallin sisäisesti että mallin rajojen yli.

Taulukko 3.7: Raportoinnin vaatimukset

Tunniste ja nimi	**Kuvaus**
Y11, Raporttien luominen	Ympäristön on tuettava asiakirjamuotoisten raporttien luomista mallidatan perusteella.
Y12, Raporttipohjat	Raporttien ulkoasun on perustuttava määriteltyihin raporttipohjiin.
Y13, Raporttipohjien luominen	Uusia raporttipohjia on pystyttävä luomaan ilman ympäristön lähdekoodin muokkaamista tai uudelleenasennusta.
Y14, Raporttipohjien tallentaminen	Raporttipohjat on tallennettava keskitetysti.

3.4.4 Tiedon tallentaminen

Edellä mainittujen vaatimusten pohjalta voidaan johtaa vaatimukset malli- ja näkymädatan tallentamiselle. Esimerkiksi jotta kaikista mallinnustyökaluista olisi mahdollista käsitellä kaikkea ympäristön sisältämää mallidataa, se on tallennettava keskitettyyn tietovarastoon. Tämä asettaa sille vaatimuksia, jotka on kuvattu taulukossa 3.8.

Taulukko 3.8: Tiedon tallentamisen vaatimukset

Tunniste ja nimi	**Kuvaus**
Y15, Tiedon tallentaminen	Ympäristön avulla luotu malli- ja näkymädata on tallennettava keskitettyyn tietovarastoon, johon kaikilla mallinnustyökaluilla on pääsy.
Y16, Malli- ja näkymätyyppien kuvausten tallentaminen	Ympäristön malli- ja näkymätyyppien kuvaukset on tallennettava samaan tietovarastoon kuin malli- ja näkymädata, mutta siitä erillään.

4. TRINITY-MALLINNUSYMPÄRISTÖ

Trinity on Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksella kehitetty joustava ja heterogeeninen hajautettu mallinnusympäristö, joka on rakennettu samannimisen integrointialustan päälle. Tässä luvussa mallinnusympäristö esitellään, kerrotaan kuinka se täyttää luvussa 3.4 esitetyt vaatimukset, kerrotaan sen mallinnustyökaluille tarjoamista palveluista ja lopulta esitellään sen tekstinkäsittelyohjelmistopohjaiselle mallinnustyökalulle asettamat vaatimukset.

4.1 Trinityn yleiskuvaus

Tässä luvussa kuvataan Trinityn perusperiaatteet ja tämän työn kannalta tärkeimmät piirteet. Tärkeimmät piirteet voidaan jakaa ympäristön rakenteeseen, mallidatan tallennustapaan ja ympäristön sisällä yleisesti käytettyihin mekanismeihin.

4.1.1 Trinity-ympäristön perusperiaatteet

Trinity perustuu muutamaan perusperiaatteeseen, joiden ympärille se on suunniteltu. Nämä perusperiaatteet ovat joustava mallintaminen, heterogeeninen mallinnusympäristö, helppo laajennettavuus, muutosten automaattinen tallentaminen, tietokantapohjaisuus sekä käyttäjien samanaikaisen työskentelyn tukeminen.

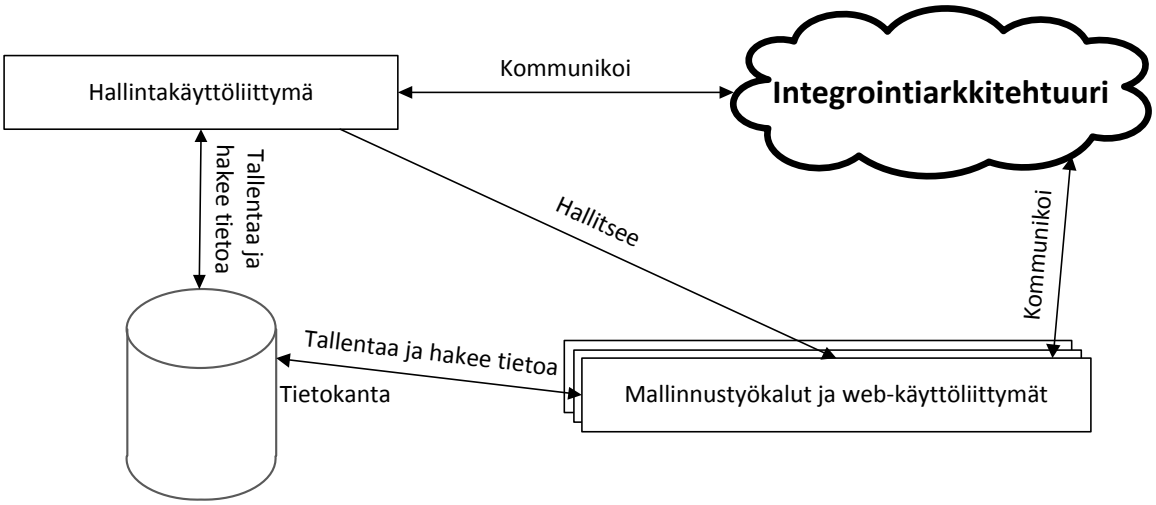
Joustavalla mallintamisella tarkoitetaan sitä, että ympäristö mukautuu käyttäjiensä työtapoihin ja käyttötilanteisiin sen sijasta, että se määrittäisi itse jäykät toimintamallit, joiden pohjalta käyttäjien tarvitsisi muokata omaa työskentelyään. Tätä tukee myös **heterogeenisen mallinnusympäristön** periaate: mallinnustietoa voidaan syöttää ja esittää monella eri työkalulla (**Y01**, **Y03**, **Y04**). Käyttäjien valmiiksi osaamia työskentelytapoja tuetaan integroitumalla valmiiksi käytössä oleviin ja käyttäjien valmiiksi tuntemiin ohjelmistoihin, kuten Microsoft Officeen. Ympäristö tukee myös useita mallinnuskieliä, ja sitä voidaan **laajentaa helposti** tukemaan uusia mallinnuskieliä ja -työkaluja (**Y02**).

Muutosten automaattisen tallentamisen, **tietokantapohjaisuuden** ja **samanaikaisen työskentelyn** periaatteet liittyvät kiinteästi toisiinsa. Mallidata tallennetaan tiedostojen sijasta tietokantaan, josta se on helposti kaikkien ympäristön käyttäjien saatavilla (**Y15**). Ympäristöllä voi olla monta yhtäaikaista käyttäjää, jotka kaikki voivat muokata samaa mallidataa samanaikaisesti. Malliin tehdyt muutok-

set tallennetaan automaattisesti tietokantaan, ja välitetään sieltä lähes välittömästi ympäristön muille käyttäjille.

4.1.2 Ympäristön rakenne

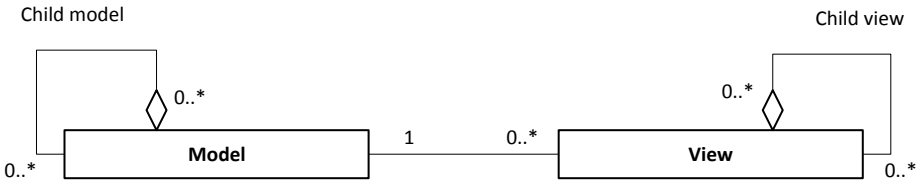
Trinity koostuu tietokannoista, hallintakäyttöliittymästä ja joukosta mallinnustyökaluja. Sen mallinnustyökaluihin kuuluu tässä työssä kuvatun Microsoft Word -pohjaisen mallinnustyökalun lisäksi muun muassa kaaviomuotoisen tiedon syöttämiseen ja esittämiseen tarkoitettu Microsoft Visio -pohjainen työkalu [2], joukko tiettyihin käyttötarkoituksiin erikoistettuja www-selainpohjaisia käyttöliittymiä sekä Microsoft Excel -pohjainen työkalu. Microsoft Excel -työkalu on kehitetty rinnakkain Word-työkalun kanssa, ja niiden arkkitehtuuri on osittain yhteinen. Kuvassa 4.1 on esitetty ympäristön arkkitehtuuri korkealla tasolla.



Kuva 4.1: Trinity-mallinnusympäristön korkean tason arkkitehtuuri.

4.1.3 Malli- ja näkymädatan tallentaminen

Mallidata on jaettu Trinityssä malleihin, jotka edustavat mallinnettavaa kohdetta. Yksittäinen näkymä voi kuvata esimerkiksi yksittäistä luokkakaaviota tai dokumentin lukua. Näkymillä on tyyppi, joka määrittelee minkä tyyppisiä näkymäelementtejä niihin voi luoda ja millä mallinnustyökaluilla se voidaan avata. Kukin malli liittyy johonkin ympäristön mallinnuskieleen, joka määrittelee minkä tyyppisiä näkymiä ja mallielementtejä ne voivat sisältää (**Y07**). Esimerkiksi Activity Model -mallityypin malli sisältää aktiviteettikaavioita. Myös mallinnuskielten ja näkymätyyppien kuvaukset on tallennettu ympäristön keskitettyyn tietokantaan, joten uusien kielten ja näkymätyyppien lisääminen on helppoa (**Y16**, **Y09**). Mallien ja näkymien välinen suhde on kuvattu kuvassa 4.2.



Kuva 4.2: Mallien ja näkymien suhde Trinityssä.

Trinityssä malli- ja näkymädata on erotettu toisistaan *malli- ja näkymäelementteihin* (**Y05**). Trinityssä yksittäinen mallielementti kuvaa sen ominaisuudet mallin suhteen, mutta sillä ei ole graafista esitysmuotoa. Esimerkkejä mallielementeistä ovat luokkakaavion luokat ja käyttötapauksen askeleet. Näkymäelementti on mallielementin graafinen esitysmuoto, ja sen rakenne on erilainen joka mallinnustyökalussa (**Y06**).

Kukin mallielementti voidaan kuvata useammalla näkymäelementillä, ja kullekin mallielementin esiintymälle luodaan oma näkymäelementti. Esimerkiksi jos sama askel esiintyy kahdessa eri käyttötapauksessa, molemmille esiintymille luodaan oma näkymäelementti. Saman mallielementin voi myös kuvata usealla eri näkymäelementtityypillä, esimerkiksi sama käyttötapauksen askel voidaan kuvata Visio-laajennoksessa graafisesti ja Word-laajennoksessa tekstuaalisesti.

Yksi mallielementtiä kuvaavista näkymäelementeistä on sen *ensisijainen näkymäelementti* (primary view element), ja muut ovat *viittaavia näkymäelementtejä* (reference view element). Ensisijaisen näkymäelementin tarkoitus on sijaita siinä näkymässä, jossa kyseinen mallielementti on ensimmäisenä määritelty. Siihen näkymään, jossa se sijaitsee, on mahdollista navigoida helposti kyseisen mallielementin viittaavista näkymäelementeistä. Jos ensisijainen näkymäelementti tuhotaan, jostain muusta kyseiseen mallielementtiin liittyvästä näkymäelementistä tehdään sille uusi ensisijainen näkymäelementti.

Lapsielementit esitetään isäelementin sisällä *lokeroelementtien* (compartment element) avulla. Lokeroelementit ovat näkymäelementtien lapsielementteinä olevia näkymäelementtejä, jotka kokoavat yhteen ja ryhmittelevät isäelementtiin elementtipohjassaan määritellyllä suhteella liittyviä mallielementtejä. Esimerkkinä lokeroelementistä toimii luokkakaavion luokan tapa kuvata siihen liittyvät operaatiot. Isäelementille on yleensä mahdollista luoda uusia lapsielementtejä lokeroelementtien kautta.

4.1.4 Elementtipohjat

Näkymäelementtien alkuperäiset ominaisuudet ja ulkoasu perustuvat *elementtipohjiin* (template element). Ne kuvaavat näkymäelementtien *metaominaisuudet*, kuten mihin mallielementtityyppiin ne liittyvät ja minkälaisia lapsielementtejä niille voi

liittää. Pohjat eivät pakota näkymäelementtien ulkoasua vastaamaan niitä, vaan määrittävät ainoastaan miltä ne näyttävät heti luomisen jälkeen.

Elementtipohjat ovat mallinnuskielten ohella yksi ympäristön tärkeimmistä laajennusmekanismeista. Muokkaamalla olemassaolevia ja luomalla uusia elementtipohjia voidaan sekä ylläpitää olemassaolevia mallinnuskieliä vastaamaan niihin tulleita muutoksia että luoda uusille kielille elementtipohjia, joiden avulla kieltä voi käyttää mallinnukseen. Elementtipohjien tarkempi rakenne on esitelty luvussa 5.3.

4.1.5 Suhteet ja näkymähierarkiat

Trinityn keskeiset käsitteet, kuten mallit, näkymät, malli- ja näkymäelementit yhdistetään toisiinsa mallinnuskielessä määriteltyjen *suhteiden* avulla. Mallielementtien väliset viittaukset toteutetaan niiden avulla, esimerkiksi käyttötapauksen askelien väliset siirtymät yhdistetään askeliin niiden avulla. Niitä voidaan luoda suhteen tyypistä riippuen myös minkä tahansa edellä mainittujen käsitteiden välille, esimerkiksi mallielementti voi liittyä malliin. Samoin esimerkiksi mallielementillä voi olla suhde toiseen mallielementtiin toisessa mallissa (**Y10**). Suhteilla voi olla myös järjestysnumero, joka ilmaisee niiden keskinäisen järjestyksen suhteessa isäkäsitteeseen.

Suhteiden tämän työn kannalta tärkein käyttökohde on näkymien yhdistäminen toisiinsa näkymähierarkioiksi (**Y08**). Tähän käytetään isä-lapsi-suhteita. Näkymien keskinäinen järjestys kullakin hierarkian tasolla suhteessa isänäkymäänsä määritellään järjestysnumeroilla.

4.1.6 Kaikille mallinnustyökaluille yhteiset toimintatavat ja mekanismit

Ympäristö määrittelee kaikille sen mallinnustyökaluille yhteisiä toimintatapoja ja mekanismeja. Tällaisia ovat esimerkiksi värjäystilat, erilaiset tavat kopioida ja liittää mallielementtejä sekä ympäristön yleinen viittaustapa.

Värjäystilat ovat ympäristön tapa esittää näkymiin liittyvää tietoa, joka on käyttäjän kannalta hyödyllistä, mutta joka ei ole puhdasta malli- tai näkymädataa. Ne perustuvat näkymäelementtien värjäämiseen näkymässä. Esimerkki värjäystilasta on *Muuttuneet elementit* -värjäystila, joka värjää näkymän avaamisen jälkeen siihen luodut uudet näkymäelementit ja avaamisen jälkeen muokatut elementit. Kukin väritystila pyritään esittämään kaikissa ympäristön mallinnustyökaluissa samalla värillä. Esimerkiksi Muuttuneet elementit -värjäystilan ollessa päällä uudet elementit näytetään vihreällä ja muuttuneet elementit sinisellä.

Trinity-ympäristö määrittää neljä koko ympäristön laajuista tapaa **kopioida elementtejä** näkymien sisällä ja välillä. Käytettävä kopiointitapa päätetään vasta liittämisoperaatin yhteydessä, sillä sen käyttötilanne vaikuttaa merkittävästi siihen, mi-

kä kopiointitapa valitaan. Ympäristön määrittämät tavat ovat: *Täyskopiointi*, jossa sekä kopioitavista näkymäelementeistä että niiden mallielementeistä luodaan kopiot kohdemalliin ja -näkymään. Luodut näkymäelementit asetetaan luotujen mallielementtien ensisijaisiksi näkymäelementeiksi. *Viittauskopioinnissa* kopioitavista näkymäelementeistä luodaan kopiot kohdenäkymään. Luodut näkymäelementit asetetaan osoittamaan alkuperäisiin mallielementteihin.

Ympäristö määrittää myös kaksi edellisiin kopiointimekanismeihin perustuvaa kopiointitapaa. *Normaali kopiointi* tekee joko täyskopioinnin tai viittauskopioinnin kopioitavien näkymäelementtien perusteella. Ensisijaisille näkymäelementeille suoritetaan täyskopiointi ja viittaaville näkymäelementeille suoritetaan viittauskopiointi. *Siirtäminen* tekee kopioitavalle elementille täyskopioinnin, mutta alkuperäiset elementit poistetaan. Tämän vaikutuksesta elementit käytännössä siirtyvät kohdenäkymään. Tämä on ympäristön vastine leikkaa ja liimaa -toiminnolle.

Trinity määrittelee **ympäristön yleisen viittaustavan** URI:en avulla. Sen avulla voidaan sekä jakaa linkkejä esimerkiksi johonkin näkymiin että suorittaa toimintoja ympäristössä. URI:t koostuvat protokollaosasta, komento-osasta ja sen parametreista seuraavan BNF:n mukaisesti:

<uri> ::= <protokolla>:<komento>?<parametrit>

<parametrit> ::= <parametri>=<arvo>[|<arvo>]*[\&<parametrit>]*

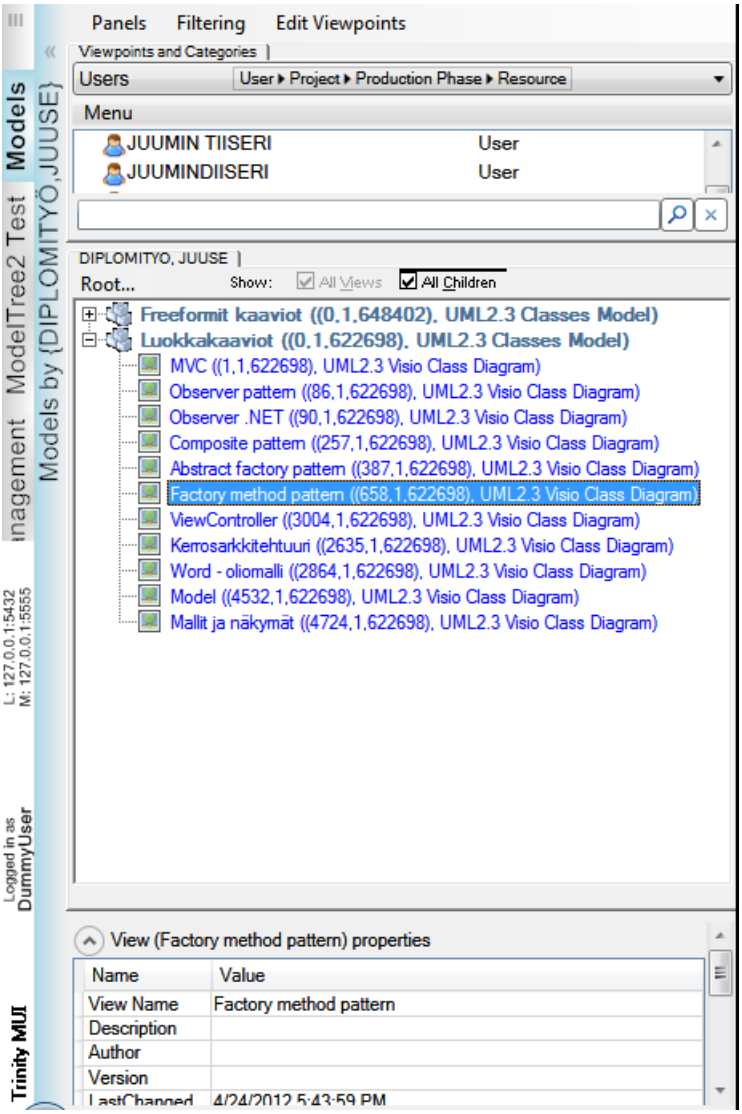
URI:en protokollaosana käytetään aina merkkijonoa *trinity*. Komento-osan tämän työn kannalta keskeisimmät arvot ovat *openview* ja *paste*. Openview-komennon avulla voidaan luoda linkkejä näkymiin. Sen parametrina annetaan näkymä, joka avataan, kun kyseistä URI:a seurataan. Paste-komentoa käytetään kopioitaessa elementtejä mallinnustyökalusta toiseen. Kun johonkin auki olevaan näkymään esimerkiksi kopioidaan ja liitetään paste-komennon sisältävä URI, siihen kopioidaan ja liitetään URI:n parametrina annetut elementit.

4.2 Ympäristön tarjoamat palvelut

Trinity tarjoaa työkaluilleen joukon koko ympäristölle yhteisiä komponentteja ja palveluita. Näistä tärkeimmät ovat edellä mainittu ympäristön hallintaan tarkoitettu *hallintakäyttöliittymä*, ja mallinnustiedon näyttämiseen tarkoitettu *informaatiopaneeli*. Niistä kerrotaan tarkemmin luvuissa 4.2.1 ja 4.2.2.

4.2.1 Hallintakäyttöliittymä

Hallintakäyttöliittymä (MUI, Management User Interface) on mallinnusympäristön hallintaan tarkoitettu työkalu. Päällä ollessaan se on aina näkyvillä, ja siihen pääsee käsiksi ruudun vasemmasta reunasta. Sillä on kaksi tilaa, *pienennetty* ja *laa-*



Kuva 4.3: Trinity-ympäristön hallintakäyttöliittymä laajennetussa tilassa, Luokkakaaviot-mallin ViewController-näkymä valittuna.

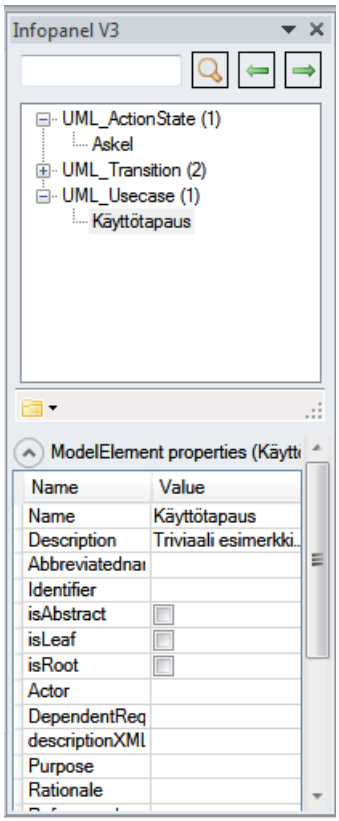
jennettu. Pienennetyssä tilassa se näytetään ohuena, pystysuorana palkkina, jolloin se näyttää tilainformaatiota ympäristöstä. Laajennetussa tilassa sivupalkin oikealle puolelle aukeaa joukko paneeleita, joiden kautta työkalulla tehtävät toiminnot suoritetaan. Hallintakäyttöliittymän ulkoasu laajennetussa tilassa on esitetty kuvassa 4.3. Hallintakäyttöliittymän tärkeimmät tehtävät tämän työn kannalta ovat:

- Mallien ja näkymien elinkaaren hallinta
- Mallien ja näkymien ominaisuuksien hallinta
- Näkymien avaaminen mallinnustyökaluissa
- Kirjanpito auki olevista näkymistä
- Mallien ja näkymien linkittäminen toisiinsa malli- ja näkymähierarkioiksi

Hallintakäyttöliittymässä on myös muita ominaisuuksia, kuten mallien ja näkymien kategorisointi sekä mallin rakenteen selaaminen Model Viewer -paneelin kautta. Hallintakäyttöliittymän avulla on myös mahdollista muokata Trinityn asetuksia, kuten järjestelmän mallinnustyökalujen asennuspolkuja ja tietokantojen tietoja, sekä tehdä järjestelmän huoltoon liittyviä toimenpiteitä, kuten alustaa paikallinen tietokanta. Se sisältää myös informaatiopaneelin, jonka kautta on mahdollista muokata valittujen mallien ja näkymien tietoja. Hallintakäyttöliittymän tarkempi määrittely on kuvattu lähteessä [5].

4.2.2 Informaatiopaneeli

Informaatiopaneeli on käyttöliittymäkomponentti, jonka tarkoitus on näyttää lisätietoa valituista mallielementeistä. Se esittää valittujen elementtien kaikki tiedot, mukaanlukien ne, joiden esittäminen esimerkiksi osana luokkakaaviota tai käyttötapauksen tekstuaalista kuvauksta ei olisi järkevää. Sen kautta on myös mahdollista navigoida malli- ja näkymäelementtien välillä, muokata niiden tietoja sekä luoda ja poistaa mallielementtien lapsielementtejä. Informaatiopaneelin ulkoasu on esitetty kuvassa 4.4.



Kuva 4.4: Trinityn Informaatiopaneeli esittämässä käyttötapausta kuvaavan mallielementin tietoja.

Informaatiopaneeli koostuu useasta perusikkunasta, joista jokainen kuvaa valittua mallielementtiä tai kyseisen elementin sisältävää mallia eri näkökulmasta. Perusikkunoista tämän työn kannalta oleelliset ovat *Element Information* ja *Elements*. Element Information kuvaa valittujen malli- ja näkymäelementtien tietoja taulukkomuotoisesti. Sen kautta on mahdollista muokata kyseisten elementtien tietoja ja luoda niille lapsielementtejä. Mallielementtiselain Elements kuvaa valitut mallielementit sisältävän mallin mallielementtejä puumaisena, elementin tyypin mukaan kategorisoituna rakenteena. Sen kautta on mahdollista valita elementtejä muualla mallissa, ja siitä on mahdollista raahata ja pudottaa mallielementtejä auki olevaan näkymään ja ympäristön muihin työkaluihin.

Paneeli sisältää myös muita perusikkunoita, kuten mallinnustiedon kategorisoimiseen tarkoitetun *Tags-ikkunan* sekä katselmointien hallintaan tarkoitetun *Reviews-ikkunan*. Niitä ei kuitenkaan käsitellä tässä työssä.

4.3 Trinity-ympäristön työkalulle asettamat vaatimukset

Tärkeä mallinnusympäristön osana toimivalle mallinnustyökalulle asetettava vaatimus on sen sulava integroituminen mallinnusympäristöön. Tässä luvussa kuvatut vaatimukset liittyvät kiinteästi käytettyyn mallinnusympäristöön, sen perusperiaatteiden tukemiseen ja perustoiminnallisuuksien toteuttamiseen. Vaatimusten tunnisteen "T"-etuliite tulee sanasta "Trinity".

4.3.1 Joustava mallintaminen

Jotta joustavan mallinnuksen asettamat vaatimukset olemassaolevien työtapojen tukemisesta täyttyisivät, mallinnusominaisuudet on integroitava Wordiin mahdollisimman löyhästi. Laajennoksen olemassaolo ei esimerkiksi saa vaikuttaa haitallisesti Wordin toimintaan mallinnuskäytön ulkopuolella, eikä muuttaa sen perustoiminnallisuuksia ja olemassaolevaa käyttöliittymää. Koska joustavan mallinnuksen periaatteiden mukaan Trinityn mallit voivat olla myös välillä syntaktisesti rikki, työkalun on pystyttävä kuvaamaan rikkinäistä mallidataa. Joustavan mallintamisen työkalulle asettamat vaatimukset on kuvattu taulukossa 4.1.

4.3.2 Tietomalli ja käyttäjien samanaikainen työskentely

Käyttäjien samanaikaisen työskentelyn, tietokantapohjaisuuden ja automaattisen tietojen tallentamisen vaatimukset liittyvät kiinteästi ympäristön tietointegraatioon. Jotta samanaikainen käyttö olisi mahdollista, kaikki käyttäjän tekemät muutokset täytyy tallentaa välittömästi ympäristön tietovarastoon, ja muiden käyttäjien tekemät muutokset on haettava tietovarastosta ja näytettävä näkymässä välittömästi. Työkalun on myös tuettava ympäristön tietomallia, jotta sillä voidaan näyttää

Taulukko 4.1: Joustavan mallintamisen vaatimukset

Tunniste ja nimi	Kuvaus
T01, Mallinnusominaisuuksien kytkeytyminen päälle	Mallinnusominaisuuksien on kytkeydyttävä päälle vain tarvittaessa.
T02, Perustoiminnallisuuksien säilyttäminen	Mallinnusominaisuudet eivät saa muuttaa Wordin perustoiminnallisuuksia tai olemassaolevaa käyttöliittymää mallinnusominaisuuksien ollessa pois päältä.
T03, Työskentelytapojen tukeminen	Mallinnusominaisuudet on integroitava Wordiin siten, että ne tukevat Wordin olemassaolevia ja käyttäjien tuntemia työskentelytapoja.
T04, Käyttöliittymän säilyttäminen samanlaisena	Wordin käyttöliittymä on pyrittävä säilyttämään mahdollisimman muuttumattomana myös mallinnusominaisuuksien ollessa päällä.
T05, Rikkinäiset mallit	Työkalun avulla on pystyttävä luomaan ja käsittelemään sellaista mallidataa, joka ei ole käytetyn mallinnuskielen syntaksin mukaista.

muillakin ympäristön työkaluilla luotua mallidataa. Kollaboratiivisen työskentelyn ja ympäristön tietomallin tukemisen vaatimukset on kuvattu taulukossa 4.2.

Taulukko 4.2: Ympäristön tietomallin ja kollaboratiivisen työskentelyn vaatimukset

Tunniste ja nimi	Kuvaus
T06, Tietomalli	Työkalun on tuettava ympäristön tietomallia.
T07, Muutosten tallentaminen	Työkalun kautta tehdyt muutokset on tallennettava automaattisesti ja välittömästi ympäristön tietovarastoon.
T08, Muutosten hakeminen	Muiden ympäristön käyttäjien tekemät muutokset on haettava ja näytettävä automaattisesti ja välittömästi.

4.3.3 Muut vaatimukset

Trinity-ympäristö asettaa työkalulle myös muita vaatimuksia, jotka eivät kuulu mihinkään edellä mainituista kategorioista. Ne liittyvät ympäristön yleisten toimintamallien toteuttamiseen ja Wordin erityisominaisuuksien hyödyntämiseen. Näkymiin liittyvää lisäinformaatiota tulee pystyä näyttämään värjäystilojen avulla. Dokumenttimaiset rakenteet esitetään Trinity-ympäristössä näkymähierarkioina, ja täten ne tulee pystyä esittämään työkalussa dokumentteina. Mallia kuvaavat dokumentit sisältävät usein mallidataa sisältäviä kuvia, kuten esimerkiksi luokkakaavioita. Muiden työkalujen avulla luotuja näkymiä tulee pystyä näyttämään sellaisenaan osana näkymähierarkioita. Nämä vaatimukset on esitetty taulukossa 4.3.

Taulukko 4.3: Muut Trinityn laajennokselle asettamat vaatimukset

Tunniste ja nimi	**Kuvaus**
T09, Väritykset	Työkalun tulee tarjota värityksiä havainnollistamaan mallidataan liittyvää lisäinformaatiota.
T10, Näkymähierarkioiden esittäminen	Ympäristön hallintatyökalulla luodut näkymähierarkiat tulee pystyä esittämään asiakirjoina.
T11, Muille työkaluille määriteltyjen näkymien esittäminen	Muille työkaluille määritellyt näkymät on esitettävä sellaisenaan.

5. WORDIN LAAJENTAMINEN MALLINNUSTYÖKALUKSI

Tässä työssä suunniteltiin ja toteutettiin Microsoft Word -laajennos, *Trinity Word Add-in* (laajennos, Word-laajennos), joka muuntaa Wordin tekstinkäsittelypohjaiseksi ohjelmistojen mallinnustyökaluksi Trinity-ympäristöön. Tässä luvussa esitellään laajennoksen käyttöliittymä ja tärkeimmät toiminnot, ja yhdistetään ne luvuissa 3.3 ja 4.3 esiteltyihin vaatimuksiin.

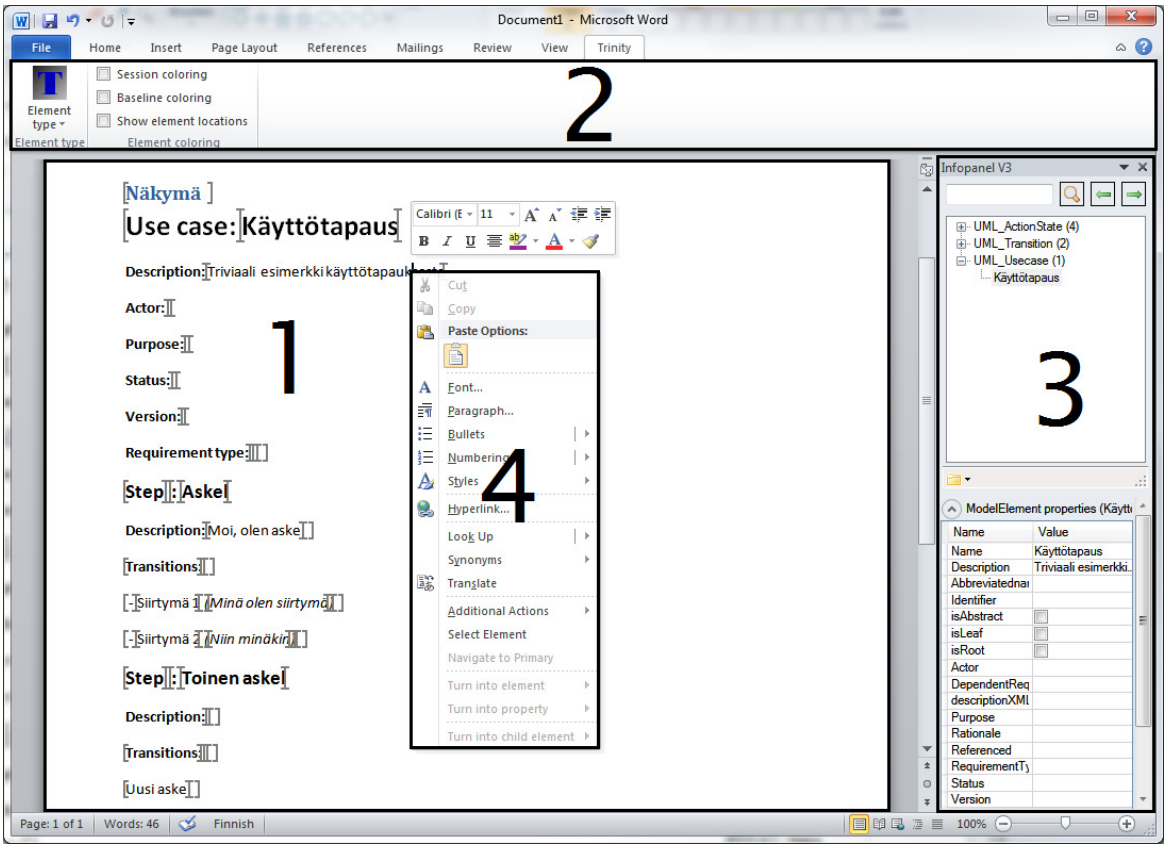
5.1 Käyttöliittymä

Trinity Word Add-inin toiminnallisuudet on integroitu Wordiin mahdollisimman huomiota herättämättömästi. Esimerkiksi laajennoksen mallinnusominaisuudet kytketään päälle ja tuodaan näkyviin vasta siinä vaiheessa kun käyttäjä avaa ensimmäisen näkymän (**T01**). Laajennos ei muuta Wordin olemassaolevia työkaluja ja komentoja mitenkään, vaan ainoastaan lisää niiden rinnalle uusia (**T02**).

Kuvassa 5.1 on esitelty laajennoksen käyttöliittymä. Kuvaan on merkitty sen tärkeimmät osat. Laajennos muokkaa Wordin tavallista käyttöliittymää kolmella tavalla: luomalla valintanauhaan *Trinity-välilehden*, luomalla *kontekstivalikkoon* uusia toimintoja ja alivalikoita sekä lisäämällä Trinityn *informaatiopaneelin* Word-ikkunan oikeaan reunaan. Näistä Wordin valintanauhan Trinity-välilehti on näkyvissä myös laajennoksen muiden mallinnusominaisuuksien ollessa pois päältä, ja siten se on käyttäjän kannalta näkyvin muutos. Kontekstivalikon napit ja alivalikot sekä informaatiopaneeli näytetään käyttäjälle ainoastaan laajennoksen mallinnusominaisuuksien ollessa päällä. (**T04**)

Wordin valintanauhan **Trinity-välilehden** (kuva 5.1, alue 2) kautta on mahdollista suorittaa auki olevaan dokumenttiin ja näkymään liittyviä operaatioita. Näkymän ollessa auki valintanauhan kautta voi luoda elementtejä ja säätää näkymien värjäystiloja. Jos käyttäjällä on auki joku olemassaoleva dokumentti, valintanauhan kautta on mahdollista käynnistää avoimen asiakirjan muuntaminen mallielementeiksi ja avata tämän prosessin läpi käyneitä asiakirjoja.

Trinityn informaatiopaneeli avataan ensimmäistä näkymää avattaessa ruudun oikeaan reunaan task pane -tyyppisenä ikkunana (kuva 5.1, alue 3). Se esitetään keskeltä kahteen osaan jaettuna, pystysuorana ikkunana, jossa yläpuolella näytetään Elements-ikkuna ja alapuolella Element Information -ikkuna.



Kuva 5.1: Trinity Word Add-in ja siinä auki oleva näkymä.

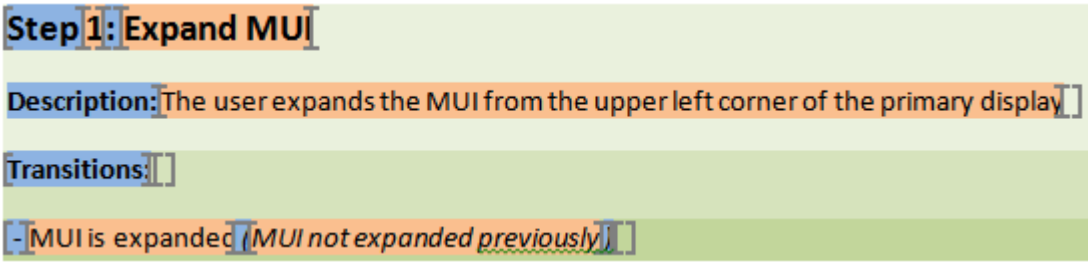
Kontekstivalikkoon (kuva 5.1, alue 4) lisättyjen toimintojen ja alivalikoiden kautta on mahdollista suorittaa operaatioita, jotka liittyvät kyseisellä hetkellä valittuihin elementteihin. Näistä esimerkkeinä ovat elementin valitseminen kokonaan, navigointi viittaavasta näkymäelementistä ensisijaiseen näkymäelementtiin.

5.2 Näkymäelementit

Mallidatan esittäminen laajennoksessa perustuu Trinityn malli- ja näkymäelementteihin, joiden merkitys ympäristössä on kuvattu luvussa 4.1.3 (**T06**). Tässä luvussa kuvataan laajennoksen käyttämien näkymäelementtien rakenne ja toiminta. Luvussa 5.2.1 esitellään näkymäelementtien rakenne ja luvussa 5.2.2 kuinka niihin liittyvien mallielementtien ominaisuudet esitetään.

5.2.1 Näkymäelementtien rakenne

Kuvassa 5.2 on esitetty käyttötapauksen askelta kuvaava näkymäelementti. Wordlaajennoksen käyttämät näkymäelementit ovat yhtenäisiä tekstinpätkiä, jotka jaetaan *osiin* tyylien ja käyttötarkoituksen perusteella. Ne eroavat siten merkittävästi esimerkiksi Visio-laajennoksessa käytetyistä graafisista näkymäelementeistä.



Kuva 5.2: Käyttötapauksen askelta kuvaava näkymäelementti, jolla on askelten välinen siirtymä lapsielementtinä. Taustavärjäykset eivät ole osa näkymäelementtejä, vaan havainnollistavat elementtien eri osia. Mukailtu lähteestä 3.

Näkymäelementit koostuvat toisiinsa liitetyistä tekstinpätkistä, *osista* (part), jotka erotellaan toisistaan tyylien, typografian ja käyttötarkoituksen perusteella. Esimerkiksi tyylin, fontin tai tekstin kursivoinnin muuttuminen elementin keskellä tarkoittaa, että edellinen osa loppuu ja uusi alkaa muutoskohdasta. Samoin jos joku elementin kohta kuvaa jotain mallielementin ominaisuutta, se eriytetään omaan osaansa. Osien rajat on esitetty kuvassa 5.2 hakasulkuina ja osien käyttötarkoitukset taustavärillä. Aukeava hakasulku tarkoittaa osan alkua ja sulkeutuva hakasulku sen loppua. Näkymäelementin sisältämä puhdas teksti on kuvattu sinisellä taustalla. Mallielementtien ominaisuudet on kuvattu oranssilla taustalla.

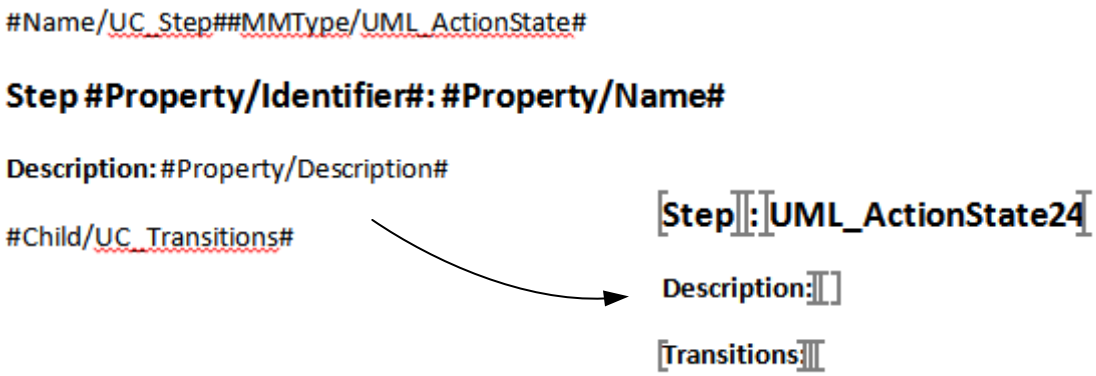
Laajennoksessa käytetään elementtihierarkioiden luomiseen luvussa 4.1.4 kuvattua Trinityn lokeroelementtimekanismia. Lapsielementit esitetään isäelementissä osien avulla. Jokainen lapsielementti on isäelementin rakenteen kannalta yksi sen osa. Lapsielementit esitetään asiakirjassa isäelementtiensä sisällä olevina, omista osistaan koostuvilta mallielementeiltä. Kuvassa 5.2 on esitetty esimerkki erään askeleen muodostamasta elementtihierarkiasta vihreän taustavärin avulla: Vaalein vihreä kuvaa itse askelta. Keskimmäiseksi tummin vihreä kuvaa askeleen lapsielementteinä olevia ja siitä lähteviä siirtymiä yhteen keräävää lokeroelementtiä. Kaikkein tummin vihreä kuvaa siirtymää, joka on askeleen lapsielementti.

5.2.2 Mallielementin ominaisuuksien esittäminen

Mallidatan ominaisuuksien esitystapa näkymäelementeissä käyttää hyväksi Wordin tyylimekanismia. Jokaiselle näkymäelementissä esitetylle mallielementin ominaisuudelle luodaan oma merkkityyli. Wordissa tyylien nimet ovat uniikkeja, joten ominaisuuksia kuvaavat tyylit nimetään siten, että ne eivät mene sekaisin muiden tyylien kanssa. Tyylit nimetään näkymäelementin elementtipohjan ja kuvattavan mallielementin ominaisuuden mukaan seuraavasti: *<Elementtipohjan nimi>.<ominaisuuden nimi>*. Esimerkiksi käyttötapauksen kuvauksessa askeleen (elementtipohjan nimi UC_Step) nimeä kuvaavan tyylin nimeksi muodostuu *UC_Step.Name*.

5.3 Elementtipohjat

Trinity Word Add-inin käyttämät näkymäelementit perustuvat luvussa 4.1.4 esiteltyyn Trinity-ympäristön elementtipohjamekanismiin. Kuvassa 5.3 on esitetty vasemmalla käyttötapauksen yksittäisen askeleen elementtipohja ja oikella sen perusteella luotu elementti. Kaikki laajennoksen avulla luodut näkymäelementit, joihin liittyy jokin mallielementti, perustuvat johonkin elementtipohjaan. Toisin kuin esimerkiksi ympäristön Visio-laajennoksessa, uusien elementtien ulkoasu ei kaikissa tilanteissa kuitenkaan perustu siihen liittyvään elementtipohjaan. Tässä luvussa kuvataan laajennoksen käyttämien elementtipohjien rakenne ja syntaksi.

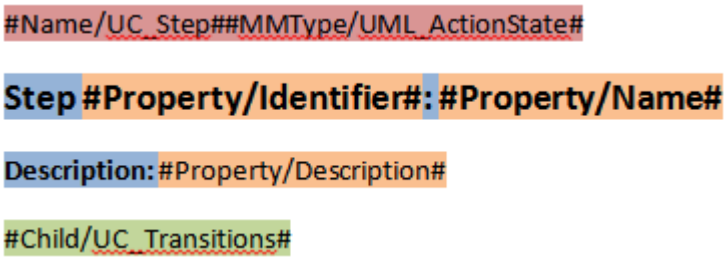


Kuva 5.3: Käyttötapauksen askeleen elementtipohja ja sen perusteella luotu näkymäelementti.

5.3.1 Elementtipohjien rakenne

Elementtipohjien rakenne on hyvin samanlainen kuin tavallisillakin näkymäelementeillä: myös elementtipohjat koostuvat yhtenäisistä tekstinpätkistä, jotka jaetaan osiin tyylien muutosten, typografisten muutosten ja käyttötarkoituksen perusteella. Kuvassa 5.4 on kuvattu käyttötapauksen askeleen elementtipohja. Kuvassa esiintyvät taustavärit eivät kuulu varsinaiseen elementtipohjaan, vaan havainnollistavat pohjassa kuvattavan tiedon kategorioita. Ne ovat: *metaominaisuudet* (kuvassa punaisella), *mallielementtien ominaisuudet* (kuvassa oranssilla ja vihreällä) ja *puhdas teksti* (kuvassa sinisellä). Niiden lisäksi osa pohjan tiedosta esitetään tyyli-informaation avulla.

Sekä metaominaisuudet että mallielementteihin liittyvät ominaisuudet kuvataan avain-arvo-pareina, joissa avaimena on ominaisuuden nimi. Ne kuvataan syntaksilla #<avain>/<arvo>#, esimerkiksi #MMType/UML_ActionState#. Arvoilla on kolme tietotyyppiä: kokonaisluku, merkkijono ja totuusarvo. Näistä kaikki kuvataan elementtipohjissa tekstillä. Mallielementteihin liittyviä ominaisuuksia on kaksi: *Property* ja *Child*. Muut avaimen saamat arvot luokitellaan metaominaisuuksiksi.



Kuva 5.4: Käyttötapauksen askeleen elementtipohja. Taustaväritykset havainnollistavat elementtipohjan eri osia, eivätkä ole osa sitä.

5.3.2 Metaominaisuudet

Metaominaisuuksien määrittämisen tarkoituksena on määrittää elementtipohjaan liittyviä ominaisuuksia, joita käytetään hyväksi elementtien luonnissa. Niitä ovat esimerkiksi pohjan perusteella luotujen näkymäelementtien kuvaamien mallielementtien tyyppi ja se, onko luotava näkymäelementti lokeroelementti. Merkittävimpiä elementtipohjan ominaisuuksia ovat *Name*, edelläkin mainittu *MMType* ja *IsCompartment*. Name määrittää elementtityypin nimen ja on ainoa ominaisuus, jolle on pakko määrittää arvo elementtipohjassa. MMType määrittää minkä tyyppiseen mallielementtityyppiin elementtipohja sidotaan. Jos sille ei anneta arvoa, elementtipohjaa ei sidota mihinkään mallielementtityyppiin.

IsCompartment kuvaa sitä, ovatko elementtipohjan perusteella luodut näkymäelementit lokeroelementtejä. Jos sille ei aseteta arvoa, se tulkitaan arvoksi False. Jos se asetetaan arvoon True, elementtipohjassa täytyy määritellä myös ominaisuudet *RelationshipName*, *ParentEndName* ja *ChildEndName*. RelationshipName määrittää suhteen nimen, jolla lokeroelementin lapsielementit kiinnitetään sen isäelementtiin. ParentEndName ja ChildEndName määrittävät, millä suhteen päillä suhde kiinnitetään isä- ja lapsielementteihin.

Osa ominaisuuksien arvoista on myös toisiaan poissulkevia. Esimerkiksi jos MMTypelle annetaan arvo, IsCompartmentille ei saa antaa arvoa True, sillä lokeroelementti ei voi kuvata mitään mallielementtityyppiä. Samoin jos IsCompartmentille ei aseteta ollenkaan arvoa tai se asetetaan arvoon False, RelationshipNamea, ParentEndNamea ja ChildEndNamea ei saa määritellä. Kullekin näkymäelementin ominaisuudelle voidaan asettaa arvo ainoastaan kerran.

Suurin osa metaominaisuuksista eivät näy käyttäjälle mitenkään lopullisissa pohjan perusteella luoduissa elementeissä. Tästä huomionarvoinen poikkeus on kuitenkin *Next*-ominaisuus, jonka avulla määritetään elementin jälkeen tuleva marginaali. Marginaalilla tarkoitetaan tässä yhteydessä tekstinpätkää, joka sijoitetaan näkymään heti elementin jälkeen ja joka määrittää miten elementit erotetaan toisistaan näkymässä.

5.3.3 Mallielementtien ominaisuudet

Mallielementteihin liittyvien ominaisuuksien on tarkoitus sitoa elementtipohjan perusteella luotuihin näkymäelementteihin liittyvien mallielementtien ominaisuuksia tekstiksi, jotta ne voidaan näyttää osana näkymää. Täten mallielementteihin liittyvät ominaisuudet näytetään osana lopullisia näkymäelementtejä.

Näkymäelementeissä näytettävät mallielementtien ominaisuudet voidaan jakaa kahteen kategoriaan: **mallielementin varsinaisiin ominaisuuksiin** (kuvassa 5.4 oranssilla) ja **lapsielementteihin** (kuvassa 5.4 vihreällä). Kuvattavan mallielementtityypin ominaisuudet kuvataan samantyyppisellä syntaksilla kuin näkymäelementteihinkin liittyvät ominaisuudet. Käytetty syntaksi mallielementtien ominaisuuksille on #Property/<ominaisuuden_nimi>#, esimerkiksi #Property/Name#. Mallielementtien ominaisuuksien määrittelyn tarkoituksena on, että siinä kohtaa varsinaista näkymäelementtiä, missä elementtipohjassa on ominaisuusmääritys, näytetään näkymäelementtiin sidotun mallielementin määritetyn niminen ominaisuus. Esimerkiksi jos käyttötapauksen askelta kuvaavan mallielementin Name-ominaisuuden arvo on *Expand MUI*, sitä kuvaavassa näkymäelementissä #Property/Name#-määritys korvattaisiin tekstillä *Expand MUI*.

Lapsielementeille syntaksi on #Child/<lapsielementtipohjan_nimi>#, esimerkiksi #Child/UC_Transitions#. Lapsielementti määrätään elementtipohjan avulla, ja kyseisen näkymäelementin lapsielementit luodaan sen pohjalta. Tämä johtuu siitä, että sama mallielementtityyppi voi olla kuvattu usealla erilaisella pohjalla, eikä pelkkä lapsielementin mallielementtityypin määrittäminen riitä yksilöimään käytettyä elementtipohjaa. Pohjan mukaisesti luoduissa näkymäelementeissä näytetään lapsielementtien elementtipohjan määrityksen syntaksin paikalla itse lapsielementti.

5.3.4 Puhdas teksti ja elementtipohjien ulkonäkö

Puhtaan tekstin avulla voidaan kuvata näkymäelementtien osia, joiden on tarkoitus pysyä samanlaisena jokaisessa elementtipohjan perusteella luodussa näkymäelementissä. Tällaisia ovat esimerkiksi käyttötapauksen askelta kuvaavien näkymäelementtien *Step*- ja *Description*-tekstit. Elementtipohjaan kirjoitettu puhdas teksti kopioidaan näkymäelementteihin sellaisenaan. Se pilkotaan osiin tyylien muutoksen ja typografisten muutosten mukaan.

Näkymäelementtityypeissä käytettyjen tyylien on tarkoitus kuvata niiden perusteella luotujen elementtien ulkoasua typografisella tasolla. Pohjien tyylit määritellään Wordin normaalilla tyyppimekanismilla. Esimerkiksi jos näkymäelementtipohja sisältää tyylillä *Normal* sanan *Description*, se luodaan myös pohjan perusteella luotaviin näkymäelementteihin tyylillä Normal. Tästä poikkeuksena ovat ne tyylit, jotka määritellään mallielementin ominaisuuksien perusteella automaattisesti.

Käytetyn tyylin lisäksi tekstistä tallennetaan myös muita typografisia tietoja, esimerkiksi käytetyn fontin nimi, fonttikoko, tieto onko teksti lihavoitu ja tieto onko teksti kursivoitu. Mallielementtien ominaisuuksia kuvaavista elementtipohjan osista tallennetaan ainoastaan typografiset tiedot.

5.4 Raporttipohjat

Laajennoksen raporttipohjamekanismi perustuu ympäristön elementtipohjamekanismiin. Raporttipohjien avulla käyttäjät voivat tehdä mallidatan perusteella sellaisia raportteja kuin he haluavat. Raporttipohjat määritellään samantyyppisellä syntaksilla kuin elementtipohjatkin. Esimerkiksi mallielementtien ominaisuudet kuvataan raporttipohjissa samalla tavalla kuin elementtipohjissakin. Se kuitenkin eroaa elementtipohjien määrittelystä seuraavasti:

Raporteilla on pystyttävä kuvaamaan yksittäisten mallielementtien lisäksi kokonaisia elementtien ja niiden välisten suhteiden muodostamista elementtihierarkioita. Kukin raporttipohja liittyy johonkin mallielementtityyppiin ja sen pohjalta voidaan muodostaa raportti yksittäiselle mallielementille, jota kutsutaan juurielementiksi, ja sen muodostamalle elementtihierarkialle. Elementtihierarkia käydään läpi raporttipohjan mukaisesti, ja kuvataan pohjassa sisäkkäisten #StartElementForeach/<Suhteen_nimi>#- ja #EndElement/#-määritysten sisällä. Niiden sisällä oleva sisältö monistetaan raporttiin kullekin kyseiseen mallielementtiin annetulla suhteella liittyvälle mallielementille.

Raportissa näytettäviä mallielementtejä voidaan suodattaa antamalla StartElementForeach-määritykselle parametriksi suhteen sen pään nimi, jolla liittyvät mallielementit kiinnittyvät suhteeseen. Tällöin raportissa näytetään ainoastaan niiden mallielementtien tiedot, jotka liittyvät aktiiviseen mallielementtiin annetun suhteen annetulla päällä. Tällä tavalla voidaan esimerkiksi suodattaa pois kaikki käyttötapauksen askeleeseen tulevat siirtymät ja näyttää ainoastaan kaikki siitä lähtevät.

5.5 Mallintaminen

Trinity Word Add-inin käyttö mallintamisessa on kuvattu tässä luvussa. Mallinnuskäyttö on laajennoksen ensisijainen käyttökohde, ja sen vuoksi siihen liittyvät toiminnot on kuvattu muita laajennoksen käyttökohteita tarkemmin. Mallinnusominaisuudet on pyritty integroimaan siten, että ne tukevat Wordin olemassaolevia työskentelytapoja mahdollisimman hyvin (**M01**, **T03**). Tässä luvussa kerrotaan myös kuinka se täyttää luvuissa 3.3 ja 4.3 sille asetetut vaatimukset. Kunkin vaatimuksen toteuttaminen on kuvattu ilmoittamalla kyseisen vaatimuksen tunniste suluissa sen toteuttavan toiminnallisuuden yhteydessä.

5.5.1 Näkymien avaaminen

Näkymien avaaminen voidaan jakaa kahteen kategoriaan: yksittäisten näkymien avaamiseen ja näkymähierarkioiden avaamiseen. **Yksittäisiä näkymiä** voidaan avata Trinity-ympäristössä monella eri tavalla. Näistä tavoista yleisimpiä ovat näkymien avaaminen hallintakäyttöliittymän malliselaimen kautta, ympäristön yleisen viittaustavan mukaisten linkkien kautta ja navigoimalla jostain muusta ympäristön työkalusta Word-näkymässä sijaitsevaan näkymäelementtiin. Näkymiä voi avata, vaikka Word ei olisikaan käynnissä. Tällöin ympäristö käynnistää sen automaattisesti ennen näkymän avaamista.

Avattuja näkymiä ei sekoiteta mahdollisesti auki oleviin muihin asiakirjoihin, vaan näytetään omassa asiakirjassaan. Ensimmäinen avattava näkymä avataan tyhjään asiakirjaan. Jos Wordissa on jo valmiiksi auki tyhjä asiakirja, näkymä avataan sinne. Muulloin sitä varten luodaan uusi asiakirja. Tämän jälkeen näkymässä olemassaoleva sisältö ladataan tietokannasta ja näytetään asiakirjassa. Myöhemmin avattavat näkymät avataan samaan asiakirjaan kuin ensimmäinen näkymä. Avatut näkymät sijoitellaan asiakirjaan peräkkäin avaamisjärjestyksessä. Täten viimeisenä avattu näkymä löytyy asiakirjan lopusta.

Laajennoksen avulla on myös mahdollista avata ympäristön hallintakäyttöliittymän avulla luotuja **näkymähierarkioita**. Tämä onnistuu avaamalla näkymähierarkian alkupisteenä toimiva Word-näkymä. Tällöin sekä alkupisteenä toimiva näkymä että sen lapsinäkymät avataan rekursiivisesti asiakirjaan ja niistä muodostetaan sisällysluettelomainen rakenne. (**T10**)

Näkymähierarkioista on mahdollista avata myös pelkkiä yksittäisiä haaroja. Tämä onnistuu valitsemalla valitsemalla hallintakäyttöliittymän mallipuusta haluttu Word-lapsinäkymä ja avaamalla se. Tällöin asiakirjaan avataan vain kyseinen näkymä ja sen lapsinäkymät. Avattavan näkymän on oltava Word-näkymä, sillä tällä hetkellä näkymät avataan aina niille määritellyillä työkaluilla riippumatta siitä, kuuluvatko ne mihinkään näkymähierarkiaan.

Näkymähierarkioita voidaan ajatella Trinity-ympäristön tapana muodostaa dokumentteja. Tästä johtuen toisin kuin yksittäiset näkymät, jokainen näkymähierarkia avataan omaan asiakirjaansa. Lapsinäkymien keskinäinen järjestys asiakirjassa suhteessa isänäkymäänsä määräytyy näkymien välisten isä-lapsi-suhteiden järjestysnumeroiden perusteella.

5.5.2 Näkymien sulkeminen

Näkymä suljetaan laajennoksessa samalla tavalla huolimatta siitä, ovatko ne näkymähierarkian osia vai yksittäisiä avattuja näkymiä. Tieto näkymän sulkeutumisesta lähetetään aina hallintakäyttöliittymälle. Sulkemiseen on kaksi erilaista tapaa: *asia-*

kirjan sulkeminen ja *näkymän sulkemiskäsky*. Jos asiakirja, johon on avattu näkymiä, suljetaan, kaikki siinä auki olleet näkymät sulkeutuvat automaattisesti. Näkymälle voidaan lähettää myös sulkemiskäsky jostain muusta ympäristön komponentista. Tämän työn kirjoitushetkellä mikään ympäristön komponentti ei kuitenkaan tue tätä.

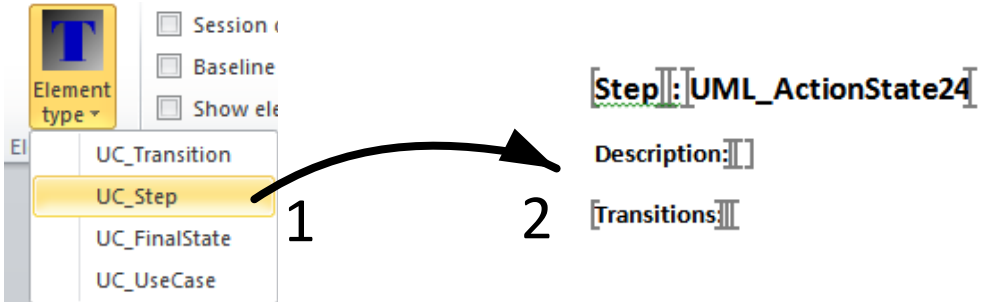
5.5.3 Elementtien luominen

Word-laajennoksessa on monta tapaa luoda uusia elementtejä, ja käytettävä tapa valitaan käyttötilanteen mukaan. Luotujen elementtien ei tarvitse liittyä toisiin elementteihin täysin mallinnuskielen syntaksin mukaisesti, esimerkiksi askelten välisen siirtymän voidaan luoda korkeimman tason elementiksi (**T05**). Elementtien luominen voidaan jakaa neljään kategoriaan: uusien ylimmän tason malli- ja näkymäelementtien luomiseen, lapsielementtien luomiseen, viittaavien näkymäelementtien luomiseen ja pelkkien näkymäelementtien luomiseen.

Uusia mallielementtejä voidaan luoda kahdella tavalla: Wordin *valintanauhan kautta* ja *annotoimalla näkymän sisältöä*. Yleisin tapa uusien mallielementtien luomiseen on Wordin valintanauhan Trinity-välilehden kautta. Siellä sijaitsevan Elementtien luonti -käyttöliittymäkomponentin (Element Creator) kautta käyttäjä voi luoda uusia malli- ja näkymäelementtejä, jotka perustuvat aiemmin määriteltyihin mallielementtipohjiin. Tällä tavalla voidaan luoda sekä ylimmän tason elementtejä että lapsielementtejä. (**M03**)

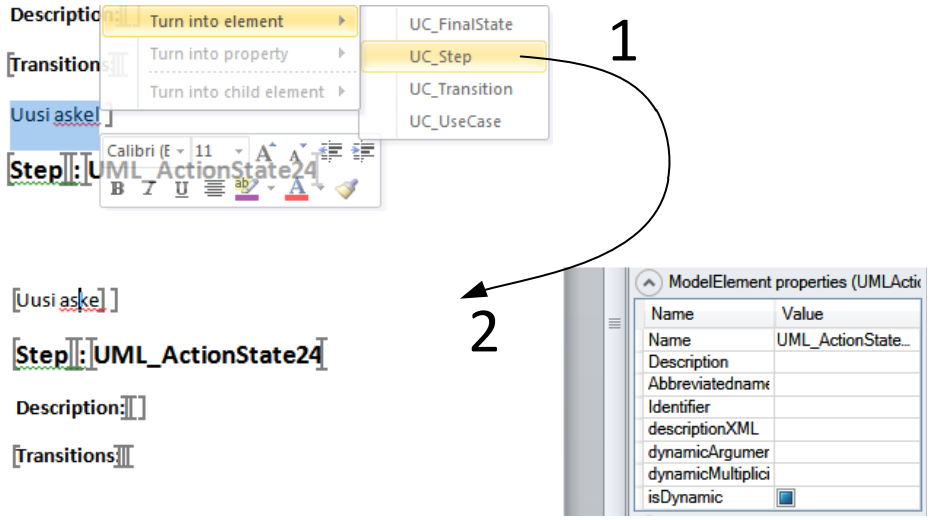
Käyttäjä valitsee pudotusvalikosta halutun mallielementtityypin. Tämän jälkeen laajennos luo näkymään sitä vastaavan, uuden mallielementin ja sitä kuvaavan, mallielementtipohjan mukaisen näkymäelementin. Luotu näkymäelementti asetetaan mallielementin ensisijaiseksi näkymäelementiksi. Jos käyttäjällä on ollut jokin yksittäinen mallielementti valittuna ennen uuden elementin luontia, uusi elementti pyritään luomaan kyseisen elementin *lapsielementiksi*. Esimerkiksi jos käyttäjällä on ollut valittuna jokin käyttötapauksen askel ja hän luo uuden askelten välisen siirtymän, siirtymä luodaan askeleen lapsielementiksi. Jos elementtiä ei pystytä luomaan lapsielementiksi, se sijoitetaan näkymässä kyseisellä hetkellä valittuna olevien näkymäelementtien perään. Esimerkkinä tästä toimii tilanne, jossa käyttäjä on valinnut jonkin käyttötapauksen askeleen ja koittaa luoda toista askelta. Askelta ei voida asettaa toisen askeleen lapsielementiksi, joten uusi askel sijoitetaan valitun askeleen perään. Esimerkki uuden ylimmän tason elementin luomisesta valintanauhan kautta on kuvattu kuvassa 5.5.

Toinen tapa uusien malli- ja näkymäelementtien luomiseen on näkymän sisältämän **tekstin annotoiminen malli- ja näkymäelementeiksi**. Tämä on mahdollista *maalaamalla* näkymästä tekstiä ja valitsemalla kontekstivalikon Turn into element -alivalikosta mallielementtityyppi, jonka tyyppiseksi elementiksi maalattu



Kuva 5.5: Uuden elementin luominen Wordin valintanauhan kautta. Kohta 1 kuvaa Elementtien luonti -käyttöliittymäkomponentin pudotusvalikkoa ja kohta 2 uutta UC_Step-elementtiä.

teksti muutetaan. Tällöin luodaan uusi mallielementti ja näkymäelementti, jonka sisältö rakennetaan maalatun tekstin pohjalta. Tällä tavalla on mahdollista luoda myös lapsielementtejä. Maalaamalla tekstiä yksittäisen mallielementtiä kuvaavan näkymäelementin sisältä ja valitsemalla kontekstivalikon *Turn into child element* -alivalikosta haluttu lapsielementin mallityyppi. Lapsielementti luodaan samoin kuin tällä tavalla luodut ylimmän tason elementitkin sillä erotuksella, että lapsielementille luodaan samalla myös lokeroelementti, johon se sijoitetaan. Esimerkki uuden ylimmän tason elementin luomisesta annotoimalla on kuvattu kuvassa 5.6. (**M04**)



Kuva 5.6: Uuden elementin luominen annotoimalla. Kohdassa 1 on valittu puhdasta tekstiä ja valittu kontekstivalikosta sen muuntaminen UC_Step-elementiksi. Kohta 2 kuvaa luotua elementtiä asiakirjassa.

Viittaavia näkymäelementtejä voidaan luoda kahdella tavalla: *raahaamalla ja pudottamalla* sekä *kopioimalla ja liittämällä* olemassaolevia mallielementtejä näkymään. Näiden toimintojen tekemisestä on kerrottu luvuissa 5.5.6 ja 5.5.7.

Tekstille, joka on kirjoitettu näkymän sisään, mutta olemassaolevien näkymäelementtien ulkopuolelle, luodaan automaattisesti oma näkymäelementti. Esimerkki

tällaisesta on tyhjään näkymään kirjoitettu teksti. Tällä tavalla luodut näkymäelementit eivät liity mihinkään mallielementtiin. (**M05**)

5.5.4 Elementtien muokaaminen

Laajennoksessa on kolme pääasiallista tapaa elementtien muokkaamiseen. Ne ovat elementin sisältävän *tekstin muokkaaminen*, mallielementin ominaisuuksien *annotointi* näkymäelementtiin ja elementin tietojen muokkaaminen *informaatiopaneelin kautta*. Kaikissa tapauksissa elementteihin tehdyt muutokset päivitetään automaattisesti tietokantaan (**T07**). Samoin kaikkien muiden käyttäjien tekemät muutokset haetaan ja näytetään näkymässä lähes välittömästi (**T08**).

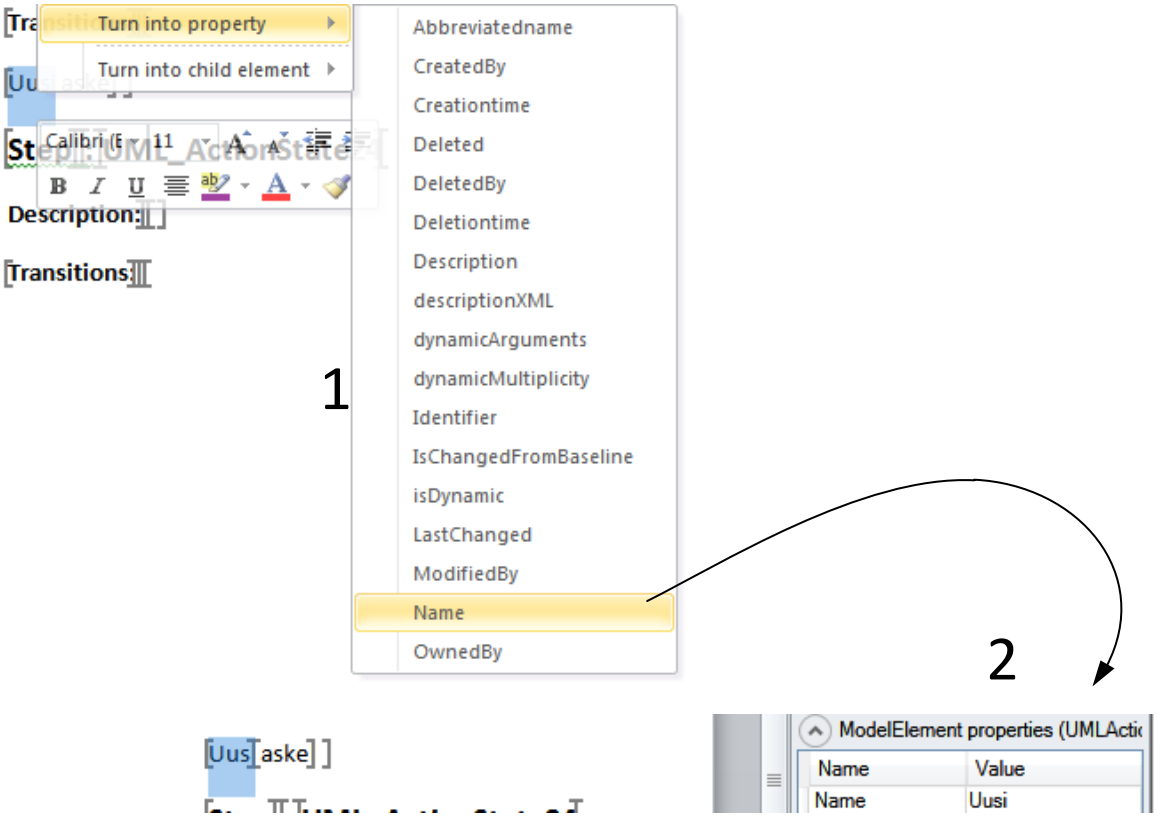
Yleisin tapa muokata elementtiä on **syöttää sinne uutta ja poistaa sieltä olemassaolevaa tekstiä**. Tämä on mahdollista kaikilla yleisesti käytetyillä tekstinkäsittelyohjelmien tiedonmuokkausmekanismeilla, kuten esimerkiksi syöttämällä ja pyyhkimällä tekstiä näppäimistön avulla sekä kopioimalla, leikkaamalla ja liittämällä tekstiä. Muutokset tallennetaan valittaessa jokin muu asiakirjan kohta tai siirrettäessä tekstinsyöttökursoria muuten kuin kirjoittamalla. Pelkästään elementin ulkoasuun tehdyt muutokset tallennetaan näkymäelementtiin. Mallielementin ominaisuuksia kuvaaviin elementin osiin tehdyt muutokset tallennetaan myös mallielementtiin. (**M02**)

Toinen tapa mallidatan muokkaamiseen on mallielementin ominaisuuksien **annotointi** mallielementteihin. Tämä on mahdollista näkymäelementeillä, joille on asetettu mallielementti. Mallielementtien ominaisuuksia voi määrittää maalaamalla näkymäelementin sisältöä ja asettamalla maalatun tekstin tyylin kyseistä mallielementin ominaisuutta kuvaavaksi tyyliksi tai avaamalla kontekstivalikon ja valitsemalla *Turn Into Property* -alavalikosta haluttu ominaisuus. Tällöin valitusta tekstistä luodaan uusi, kyseistä mallielementin ominaisuutta kuvaava näkymäelementin osa, ja kyseinen mallielementin ominaisuus asetetaan valitun tekstin sisältämään arvoon. Esimerkki tästä on esitetty kuvassa 5.7.

Kolmas tapa valittujen malli- ja näkymädatan muokkaamiseen on **mallidatan muokkaaminen informaatiopaneelin kautta**. Sen kautta on mahdollista muuttaa valittuun näkymäelementin ja siihen liittyvän mallielementin kaikkia ominaisuuksia, mukaanlukien niitä ominaisuuksia, joita ei näytetä itse näkymässä. Sen kautta ei ole mahdollista muokata näkymäelementtien ulkoasua.

5.5.5 Elementtien poistaminen

Pääasiallinen tapa elementtien poistamiseen Word-laajennoksessa on niiden pyyhkiminen kokonaan pois näkymästä, esimerkiksi valitsemalla koko elementti hiiren avulla ja painamalla näppäimistön *Delete*-painiketta. Tällöin valittu näkymäelementti



Kuva 5.7: Mallielementin ominaisuuden merkitseminen näkymäelementtiin annotoimalla. Kohta 1 kuvaa askelta kuvaavaa näkymäelementtiä, jonka sisältämä teksti halutaan annotoida elementin nimeksi. Kohta 2 kuvaa kyseistä elementtiä operaation jälkeen.

poistetaan näkymästä. Mikäli poistettu elementti oli jonkin mallielementin ensisijainen näkymäelementti, mallielementille asetetaan uusi ensisijainen näkymäelementti sitä kuvaavien näkymäelementtien joukosta. Jos poistettava näkymäelementti on ainoa kyseistä mallielementtiä kuvaava näkymäelementti, myös mallielementti poistetaan. Tällaisessa tilanteessa käyttäjältä kysytään varmistus ennen elementtien poistamista.

Lapsielementtejä voi poistaa myös informaatiopaneelin kautta. Tämä tapahtuu valitsemalla näkymästä se lokeroelementti, jossa poistettavat lapsielementit sijaitsevat, valitsemalla ne informaatiopaneelissa näytetystä lapsielementtien listasta ja painamalla lapsielementtilistan yläpuolella näytettävää *Delete*-painiketta. Käyttäjältä ei kysytä varmistusta poistettaessa elementtejä tällä tavalla.

5.5.6 Sisällön kopiointi näkymästä

Olemassaolevan sisällön kopiointi ja liittäminen on eräs Wordin normaaleista toimintamalleista uuden sisällön luomiseen (**M02**). Laajennoksen kopiointimekanismin toiminta perustuu ympäristön yleisiin, luvussa 4.1.6 kuvattuihin kopiointikäytäntöihin. Kopiointi voidaan jakaa seuraaviin kategorioihin käyttötilanteen perusteella:

kopioiminen näkymästä toiseen Word-laajennoksen sisällä, kopioiminen yksittäisen näkymän sisällä, kopioiminen muihin Word-asiakirjoihin ja kopioiminen muihin ympäristön työkaluihin.

Kopioitaessa elementtejä saman näkymän sisällä tehdään aina täyskopiointi. Tähän ratkaisuun päädyttiin, sillä sitä voidaan ajatella yhtenä tapana luoda uusia malli- ja näkymäelementtejä. Kopioitaessa elementtejä Word-näkymien välillä käyttäjälle näytetään dialogi, jonka kautta hän voi itse valita kopiointitavan ympäristön tukemista kopoitointitavoista. Kopiointioperaatio voidaan myös perua sulkemalla dialogi.

Edellä mainittujen käyttötilanteiden lisäksi laajennos määrittelee käyttäytymisen tilanteissa, jossa halutaan kopioida näkymän sisältöä muihin auki oleviin Word-asiakirjoihin ja Wordin ulkopuolelle. Kopioitaessa elementtejä Wordin sisällä muihin asiakirjoihin kopioitaan pelkkä valittu teksti. Kopioitaessa tekstiä laajennoksen Wordin ulkopuolelle kopioidaan Trinityn yleisen viittaustavan mukainen URI, jonka komento-osaksi asetetaan liittämiskomento ja parametreiksi kopioidut näkymäelementit.

5.5.7 Sisällön tuominen näkymään ulkopuolelta

Näkymiin on mahdollista tuoda sisältöä myös niiden ulkopuolelta. Tämä voidaan jakaa kahteen kategoriaan: mallidatan tuomiseen näkymiin ympäristön muista työkaluista ja tekstin tuomiseen näkymiin niiden ulkopuolelta.

Mallidatan tuominen näkymiin ympäristön muista työkaluista tapahtuu Trinityn URI-määrityksen mukaisten URI:en avulla. Kun näkymään esimerkiksi raahataan ja pudotetaan tai kopioidaan ja liitetään ympäristön URI-määrityksen mukainen URI, jonka komento-osaksi on asetettu liittämiskomento, komennon parametreina annetuille elementeille luodaan näkymään uudet viittaavat näkymäelementit. Näkymäelementit rakennetaan kyseisiä mallielementtityyppejä kuvaavien näkymäelementtipohjien perusteella. Elementit sijoitellaan näkymään samalla tavalla kuin luotaessa uusia elementtejä.

Puhtaan tekstin tuominen näkymään on mahdollista kopioimalla tekstiä esimerkiksi toisesta asiakirjasta tai web-selaimesta ja liittämällä se näkymän sisälle. Liitettyä tekstiä käsitellään samalla tavalla kuin kirjoitettuakin tekstiä. Jonkin näkymäelementin sisälle liitetty teksti liitetään osaksi kyseistä näkymäelementtiä. Näkymäelementtien ulkopuolelle, kuten esimerkiksi kahden mallielementtiä kuvaavan näkymäelementin väliin liitetylle tekstille luodaan uusi näkymäelementti, jota ei ole liitetty mihinkään mallielementtiin.

5.5.8 Värjäykset

Laajennoksesta on mahdollista kytkeä päälle värjäystiloja, joiden avulla voidaan näyttää lisäinformaatiota näkymän elementeistä (**T09**). Värjäystilan ollessa päällä osan näkymän elementeistä taustaväri muutetaan värjäystilan sisäisen logiikan mukaan. Kaikki värjäystilat esitetään eri värisillä taustaväreillä, jotta käyttäjän on helppo erottaa mitä väritystilaa kukin väri esittää. Värjäystilat asetetaan päälle Wordin valintanauhasta, ja ne koskevat aina yhtä näkymää kerrallaan. Laajennos tukee seuraavia värjäystiloja:

- Näkymän avaamisen jälkeen luotujen ja muokattujen elementtien värjääminen. Uudet elementit esitetään vihreällä ja muutetut elementit sinisellä.
- Mallin baseline-kopioinnin jälkeen muutettujen elementtien värjääminen. Esitetään näkymässä harmaalla.

5.6 Asiakirjojen muuntaminen mallidataksi

Toinen Trinity Word Add-inin toissijaisista käyttökohteista on ulkoisten dokumenttien muuntaminen ympäristöön mallidataksi (**M06**). Tässä luvussa esitellään lyhyesti siihen liittyvät toiminnot ja liitetään ne luvussa 3.3.2 kuvattuihin vaatimuksiin.

Mallidatan tuominen ympäristöön sen ulkopuolisista asiakirjoista on tarkoitettu ensisijaisesti teknisten standardien analysointiin. Asiakirjojen sisältö tallennetaan yksittäistä dokumenttia kuvaavaan Document Model -mallityypin malleihin.

5.6.1 Muuntamisprosessin suorittaminen

Ennen asiakirjan muuttamista mallidataksi ympäristössä täytyy olla luotuna tyhjä *Document Model* -mallityypin malli. Asiakirjojen muuttaminen mallidataksi tapahtuu avaamalla haluttu asiakirjatiedosto Wordissa, avaamalla Wordin valintanauhasta Trinity-välilehden ja suorittamalla sieltä *Read document into the database* -toiminto. Tämän jälkeen käyttäjälle avautuu ikkuna, josta hän voi päättää mihin malliin asiakirjan sisältö laitetaan. (**M09**)

Muunnoksen aikana analysoitavalle asiakirjalle muodostetaan oma mallielementti, joka kuvaa asiakirjaa kokonaisuutena. Se sisältää yleistä tietoa analysoitavasta asiakirjasta, kuten esimerkiksi sen nimen. Jokaiselle asiakirjan luvulle luodaan oma mallielementti, joka sisältää kyseisen asiakirjan kohdan sisällön. Esimerkiksi asiakirjan luvuille 3, 3.5 ja 3.5.3 luodaan omat mallielementit. Nämä elementit yhdistetään asiakirjaa kuvaavaan mallielementtiin suhteilla. (**M08**)

Asiakirjan lukurakennetta ylläpidetään luomalla mallielementtien välille isä-lapsi-suhteita. Esimerkiksi edellisen esimerkin tapauksessa lukua 3.5 kuvaava elementti

yhdistetään luvun 3 kuvaavan elementin lapseksi ja lukua 3.5.3 kuvaava elementti lukua 3.5 kuvaavan elementin lapseksi. Lukujen keskinäistä järjestystä ylläpidetään suhteiden järjestysnumeroiden avulla.

Mitään asiakirjan osaa ei tallenneta moneen kertaan, ja yksi mallielementti sisältää ainoastaan kuvaamansa luvun otsikon ja seuraavan otsikon välisen osuuden asiakirjasta. Esimerkiksi lukua 3 kuvaava mallielementti sisältää sen osan asiakirjasta, joka on kuvattu lukujen 3 ja 3.1 otsikoiden välissä.

5.6.2 Muunnetun asiakirjan käsittely

Koska muunnetuille asiakirjoille ei luoda näkymiä, niitä ei ole mahdollista avata hallintakäyttöliittymän kautta. Muunnetut asiakirjat voidaan avata Wordin valintanauhan Trinity-välilehden kautta. Painamalla siellä olevaa *Open document from the database* -nappia näytetään ikkuna, josta käyttäjä voi valita, mikä muunnettu asiakirja avataan. Käyttäjä voi halutessaan näyttää myös ainoastaan osan asiakirjasta valitsemalla jonkin asiakirjan luvun. Tällöin ainoastaan valittu luku ja sen aliluvut näytetään.

Muunnetun asiakirjan käsittely vastaa käyttäjän kannalta lähes täysin ympäristön ulkopuolisten asiakirjojen käsittelyä. Suurin ero tavallisten asiakirjojen käsittelyyn on se, että Trinityn perusperiaatteiden mukaisesti käyttäjän tekemät muutokset tallennetaan tietokantaan lähes välittömästi. (**M07**)

Muutetun asiakirjan lukuihin ja alilukuihin on mahdollista viitata ympäristön Visio-laajennoksesta. Tämä ominaisuus kehitettiin Visio-laajennoksen tueksi, eikä Word-näkymistä ole tällä hetkellä mahdollisuutta viitata niihin. Muunnetut asiakirjan osat ovat mallielementtejä, joten niihin voidaan viitata samalla tavalla kuin muihinkin mallielementteihin. Tämä tarkoittaa, että halutun asiakirjan lukua kuvaava mallielementti voidaan raahata ja pudottaa Visio-laajennoksessa auki olevaan näkymään esimerkiksi hallintakäyttöliittymän malliselaimesta. Tällöin käyttäjältä kysytään minkä tyyppisenä elementti näytetään Visio-näkymässä.

5.7 Raportointi

Automaattisten raporttien luonti mallidatan perusteella on toinen Trinity Word Add-inin toissijaisista käyttökohteista. Tässä luvussa esitellään lyhyesti siihen liittyvät vaatimukset ja yhdistetään ne luvuissa 3.3.3 ja 3.4.3 esiteltyihin vaatimuksiin. Koska Trinity-ympäristössä ei ole olemassa mitään keskitettyä raportointimekanismia, tässä luvussa kerrotaan myös kuinka ympäristö vastaa luvussa 3.4.3 esitettyihin vaatimuksiin. (**M10**, **Y11**)

Kaikki laajennoksen avulla luotavat raportit perustuvat raporttipohjiin, joiden rakenne on kuvattu luvussa 5.4 (**Y12**). Uuden raportin luominen käynnistetään jos-

tain ulkoisesta lähteestä, kuten esimerkiksi web-käyttöliittymästä (**M13**). Sitä ei ole mahdollista käynnistää laajennoksen itsensä kautta. Raportin luominen on käyttäjän kannalta lähes näkymätön prosessi, sillä sitä ei ole tarkoitettu suoritettavaksi käyttäjän tietokoneella (**M14**). Valmiille raportille luodaan mallielementti, johon raportin sisältö tallennetaan. Se kiinnitetään alkuperäiseen mallielementtiin suhteella. (**M11**, **M12**)

Luodut raportit tallennetaan tietokantaan Microsoft Office Open XML Document (.docx) -muotoisina tiedostoina. Tähän ratkaisuun päädyttiin, sillä raporttien on oltava katseltavissa myös sellaisilta tietokoneilta, joille ei ole asennettu Trinity-ympäristöä. Raportteja voidaan luoda ilman käyttäjän suoraa syötettä esimerkiksi palvelimella ja luomisprosessi voidaan käynnistää esimerkiksi web-käyttöliittymästä, joten on mahdollista, että osalla raportteja luovista käyttäjistä ei ole Trinity-ympäristöä asennettuna tietokoneelleen.

Koska raporteille ei luoda näkymiä, niitä ei ole mahdollista avata Trinityn hallintakäyttöliittymän kautta. Niitä ei ole mahdollista avata myöskään itse laajennoksesta. Käytännössä raportit avataan samasta käyttöliittymästä, jonka kautta ne luotiin. Tällöin raportti avataan tavallisena Word-asiakirjana. Tällainen raportti on pelkästään staattinen asiakirja, eikä sillä ole yhteyttä varsinaiseen mallidataan. Täten raportin kautta ei ole mahdollista tehdä muutoksia mallidataan, eikä raportin sisältö päivity automaattisesti mallin päivittyessä.

5.8 Ylläpitotoiminnot

Word-laajennoksella on mahdollista määritellä uusia ja muokata olemassaolevia näkymäelementti- ja raporttipohjia. Tällä tavalla saadaan lisättyä laajennoksen laajennettavuutta merkittävästi ilman, että ohjelman lähdekoodiin täytyy tehdä muutoksia. Myös joustavuus kasvaa, sillä käyttäjät voivat luoda omiin tarpeisiinsa sopivat näkymäelementti- ja raporttipohjat, eikä heidän tarvitse olla sidottuna ympäristön mukana tuleviin pohjiin. Luvussa 5.8.1 kuvataan laajennoksen tarjoamat näkymäelementtipohjien hallinnointitoiminnot ja luvussa 5.8.2 raporttipohjien hallinnointitoiminnot.

5.8.1 Elementtipohjien hallinnointi

Elementtipohjat määritellään luvussa 5.3.1 kuvatulla syntaksilla. Uusi elementtipohja määritellään syöttämällä se tyhjään asiakirjaan, valitsemalla se kokonaan ja painamalla kontekstivalikosta tai valintanauhasta Save template -painiketta. Tämän jälkeen avautuu ikkuna, josta valitaan mihin malli- ja näkymätyyppiin elementtipohja sidotaan. Painettaessa OK-painiketta elementtipohja tallennetaan tietokantaan.

Elementtipohjien muokkaaminen tapahtuu samalla tavalla, kuin uuden pohjan luominen. Jos valittuun mallityyppiin liittyy ennestään samalla nimellä oleva elementtipohja, se ylikirjoitetaan. Elementtipohjien poistaminen ei tällä hetkellä ole mahdollista. (**M15**)

5.8.2 Raporttipohjien hallinnointi

Laajennoksen tukemat raporttipohjat määritellään luvussa 5.4 kuvatulla syntaksilla. Kullekin raporttipohjalle luodaan Microsoft Office Open XML Document (.docx) -tiedosto, joka sijoitetaan laajennoksen asennushakemistoon (**Y13**), (**Y14**). Raporttipohjatiedostot nimetään vastaamaan niitä mallielementtityyppejä, joita niiden sisältämät pohjat kuvaavat. Esimerkiksi käyttötapauksen askelta kuvaavan raporttipohjan tiedostonimi olisi UML_ActionState.docx. Tästä johtuen kutakin mallielementtityyppiä voi kuvata ainostaan yksi raporttipohja.

Olemassaolevan raporttipohjan muokkaaminen onnistuu muokkaamalla sen kuvaavaa tiedostoa. Raporttipohjan voi poistaa poistamalla sen tiedoston, joka kuvaa kyseistä pohjaa. (**M16**)

5.9 Integraatio muihin työkaluihin

Word-laajennoksella on mahdollista avata myös muilla ympäristön työkaluilla luotuja näkymiä osana näkymähierarkioita. (**T11**) Tällä tavalla on mahdollista esimerkiksi luoda käyttötapauksesta havainnollistava dokumentti, joka sisältää ensin käyttötapauksen graafisen esityksen ja sen jälkeen saman esitettynä tekstuaalisessa muodossa. Toisaalta tämä mahdollistaisi myös esimerkiksi samanlaisen esityksen tekemisen luokkakaaviolle osana mallinnettavan järjestelmän määrittelydokumenttia. Laajennoksen avulla voidaan näyttää Visio- ja Excel-näkymiä.

Visio-näkymät näytetään Wordissa kuvina. Tästä johtuen niiden sisältöä ei ole mahdollista muokata suoraan Wordista. Ne kuitenkin päivitetään automaattisesti asiakirjaan jos niiden kuvaamien näkymien sisältö päivittyy. Täten esimerkiksi jos jossain auki olevassa Word-näkymässä ja Visio-näkymässä on samaan mallielementtiin viittaava näkymäelementti, ja sen tietoja päivitetään Word-näkymän kautta, myös Visio-näkymän kuva päivitetään. Kuvat luodaan Vision avulla, joten sen on oltava päällä, ja käynnistetään automaattisesti tarvittaessa.

Excel-näkymät esitetään asiakirjassa Wordin tavallisen taulukkomekanismin mukaisina taulukkoina. Tähän ratkaisuun päädyttiin, sillä Word tukee Excel-taulukoiden liittämistä osaksi asiakirjaa tällä tavalla. Taulukot luodaan Excelissä, joten sen on oltava päällä taulukon luomisen aikaan. Koska Excel-näkymät näkyvät asiakirjassa tavallisina Word-taulukkoina, ne eivät sisällä minkäänlaista takaisinkytkentää niiden kuvaamaan mallidataan. Tästä johtuen niiden kautta ei ole mahdollista muo-

kata niiden kuvaamaa mallidataa. Tämän työn kirjoitushetkellä Wordissa näytettyjen Excel-näkymien sisältämä data ei päivity automaattisesti asiakirjaan, kun sen sisältöä muokataan ulkoisesti.

6. WORD-LAAJENNOKSEN TOTEUTUS

Tässä luvussa kuvataan Trinity Word Add-inin arkkitehtuuri, suunnitteluperiaatteet ja toteutustekniikat. Tämän lisäksi luvussa kuvataan Trinity-ympäristön yleinen arkkitehtuuri sekä sen tämän työn kannalta oleelliset suunnitteluratkaisut ja ohjelmistokomponentit.

6.1 Suunnitteluperiaatteet

Luvussa 4.1.1 kuvatuista ympäristön perusperiaatteista joustavan mallinnuksen tukeminen ja helppo laajennettavuus ovat työkalun merkittäviä suunnitteluperiaatteita. Helppo laajennettavuus vaikuttaa työkalun suunnitteluun siten, että työkalu tukee uusien ominaisuuksien nopeaa toteuttamista ja integroimista.

Näiden lisäksi työkalun tärkeä suunnitteluperiaate on ohjelmistokomponenttien yleiskäyttöisyys. Työkalun komponentit pyritään toteuttamaan siten, että niitä voidaan käyttää mahdollisimman monessa paikassa, mahdollisesti ympäristön muissakin työkaluissa. Tällä tavalla identtisiä toiminnallisuuksia ei tarvitse toteuttaa moneen kertaan.

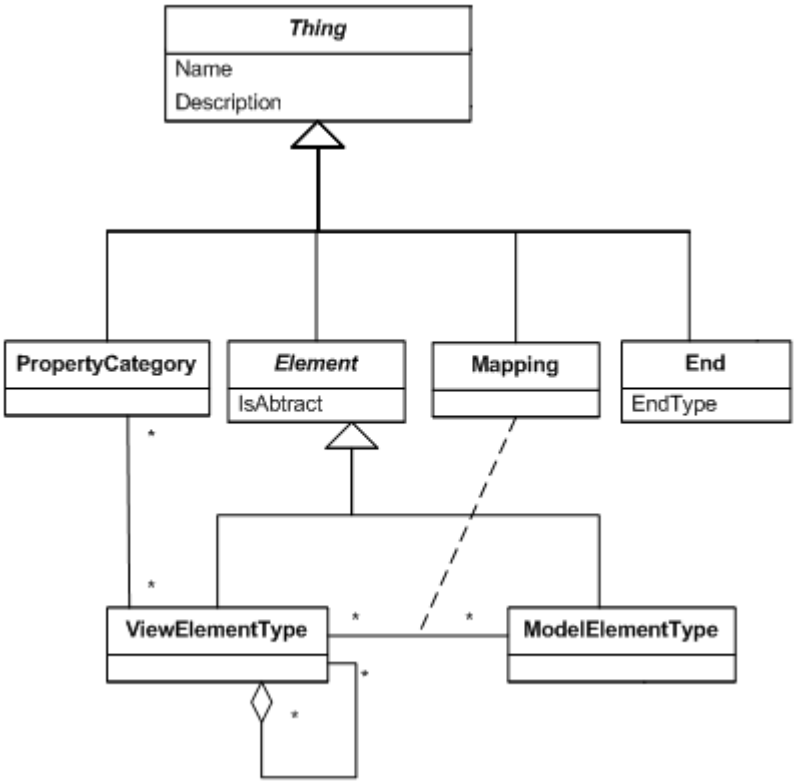
6.2 Ympäristön arkkitehtuuri

Tässä luvussa kuvataan Trinity-ympäristön arkkitehtuurin ne osat, jotka ovat Trinity Word Add-inin kannalta mielenkiintoisia. Luvussa 6.2.1 käsitellään ympäristön tietomalli, luvussa 6.2.2 tietokantakomponentti ja luvussa 6.2.3 integraatioarkkitehtuuri.

6.2.1 Tietokanta ja tietomalli

Trinity-ympäristössä kaikki data tallennetaan tietokantaan. Tietokannanhallintajärjestelmänä käytetään PostgreSQL:ää. Työkalut käsittelevät tietokannan sisältöä luvussa 6.2.2 kuvatun tietokantakomponentin kautta. Tietokannan rakenne on dynaaminen ja perustuu ympäristön kolmetasoiseen metatasohierarkiaan. Näistä tasoista ylin, *metametataso*, sisältää ympäristön *metametamallin*. Kuvassa 6.1 on ote metametamallista, koko metametamallin rakenne on kuvattu liitteessä 1. Sen avulla pystytään kuvaamaan kaikki ympäristön käsitteet, ja se sisältää abstrakteja käsit-

teitä kuten elementti, elementtien välinen suhde ja mallinnustyökalu. Metametataso on tietokannan muuttumattomin osa, ja siihen tehdään muutoksia harvoin.



Kuva 6.1: Ote Trinityn metametamallista.

Metametatason alapuolella on *metataso*, joka kuvaa ympäristön mallinnuskielet. Jokainen mallinnuskieli kuvataan omana tietokantakaavionansa. Ympäristön varsinaiset mallit luodaan näiden skeemojen rakenteen perusteella. Metatason käsitteitä ovat esimerkiksi UML 2.3:n luokkakaavio ja aktiviteettikaavio. Metataso on metametatasoa dynaamisempi ja muuttuu sitä mukaa kun ympäristön mallinnuskieliä lisätään ja muokataan.

Hierarkian alimmalla tasolla on *mallitaso*, jolla ympäristön varsinainen malli- ja näkymäinformaatio sijaitsee. Jokainen mallitason tietokantakaavio kuvaa yksittäistä mallia ympäristössä. Mallitaso on erittäin dynaaminen ja muuttuu aina mallitai näkymäinformaation muuttuessa. Esimerkkejä mallitason käsitteistä ovat UML 2.3:n luokkakaavion yksittäinen luokka, yksi askel käyttötapauksen tekstuaalisessa kuvauksessa ja teknisen standardin aliluku. Mallitasolla kuvataan myös mallien väliset yhteydet, kuten malli- ja näkymähierarkiat sekä mallielementtien viittaukset mallielementteihin toisissa malleissa, ja mallien sekä niiden sisältävän datan kategoriat.

6.2.2 Tietokantakomponentti

Tietokantakomponentti (TrinityDB, Trinity ORM) on kirjasto, joka huolehtii ympäristön tietokantayhteyksistä. Se on oliorelaatiomuunnin ja tarjoaa joukon rajapintoja, joiden kautta se voi käsitellä tietokannassa olevaa dataa. Seuraavassa on kerrottu sen Word-laajennoksen kannalta tärkeimmät ominaisuudet.

Komponentin tarjoamat rajapinnat pyritään pitämään mahdollisimman kiinteinä, ja ne muuttuvat ainoastaan, jos tietokannan metametatason rakenne muuttuu. Täten esimerkiksi jokaiselle mallielementtityypille ei ole omaa rajapintaa, vaan mallielementtien tietoja käsitellään *IModelElement*-rajapinnan kautta. Samoin malleille on *IModel*-rajapinta, näkymille *IView*-rajapinta ja näkymäelementeille *IViewElement*-rajapinta. Rajapintojen kautta on mahdollista käsitellä myös kunkin asian tyyppikohtaisia tietoja, kuten esimerkiksi mallielementin mallielementtityyppikohtaisia attribuutteja. Työkalukohtaisen näkymäinformaation käsittelyä varten tietokantakomponentti tarjoaa geneerisen *IToolSpecificInfo*-rajapinnan. Jokainen työkalu periyttää tästä rajapinnasta oman toteutuksensa. Tietokantakomponentti tarjoaa myös tapahtumia, joiden avulla työkalut voivat saada informaatiota muiden työkalujen ja käyttäjien tietokantaan tekemistä muutoksista lähes reaaliaikaisesti.

Mallidatan käsittelyn lisäksi tietokantakomponentti tarjoaa monia muitakin tietokannan hallintaan liittyviä mekanismeja mekanismeja. Näistä tämän työn kannalta keskeisimpiä ovat transaktioiden hallinta ja paikallisten tietovarastotietokantojen päätietokantaan synkronoinnin hallinta.

6.2.3 Agenttiarkkitehtuuri

Trinity-ympäristössä käytetään työkalujen integrointiarkkitehtuurina ohjelmointikieliriippumatonta agenttiarkkitehtuuria. Sen toiminta perustuu itsenäisiin *agenteihin*, jotka liikkuvat ympäristössä olevien *paikkojen* ja *alueiden* välillä suorittaen niissä *tehtäviä*. Agenttipohjaista integraatioratkaisua käyttämällä hajautettujen toimintojen toteutuslogiikkaa saadaan keskitettyä yhteen paikkaan, eli agenttiin. [14]

Agenttiarkkitehtuurin kontekstissa *alue* (Area) on kokoelma samalla tietokoneella sijaitsevia ja samalla ohjelmointikielellä ohjelmoituja *paikkoja*. Se voi myös koostua muista alueista koostesuunnittelumallin mukaisesti.

Paikat (Location) tarjoavat palveluita, joiden avulla agentit voivat suorittaa tehtäviä. Kukin paikka liittyy aina johonkin ympäristön autonomiseen kokonaisuuteen, kuten esimerkiksi tietokantaan tai mallinnustyökaluun. Täten esimerkiksi Word-laajennoksella on oma paikka.

Tehtävät (Task) ovat agentin suorituksen perusyksikköjä. Jokainen tehtävä on sidottu johonkin paikkaan, ja käyttää paikan sille tarjoamia palveluita. Tehtävällä on myös tulos, joka määräytyy sen perusteella, saatiinko tehtävä suoritettua kunnolla.

Agenttien suoritus voi haarautua tehtävän tuloksen perusteella. Esimerkki Word-laajennoksen toteuttavasta tehtävästä on näkymän avaaminen.

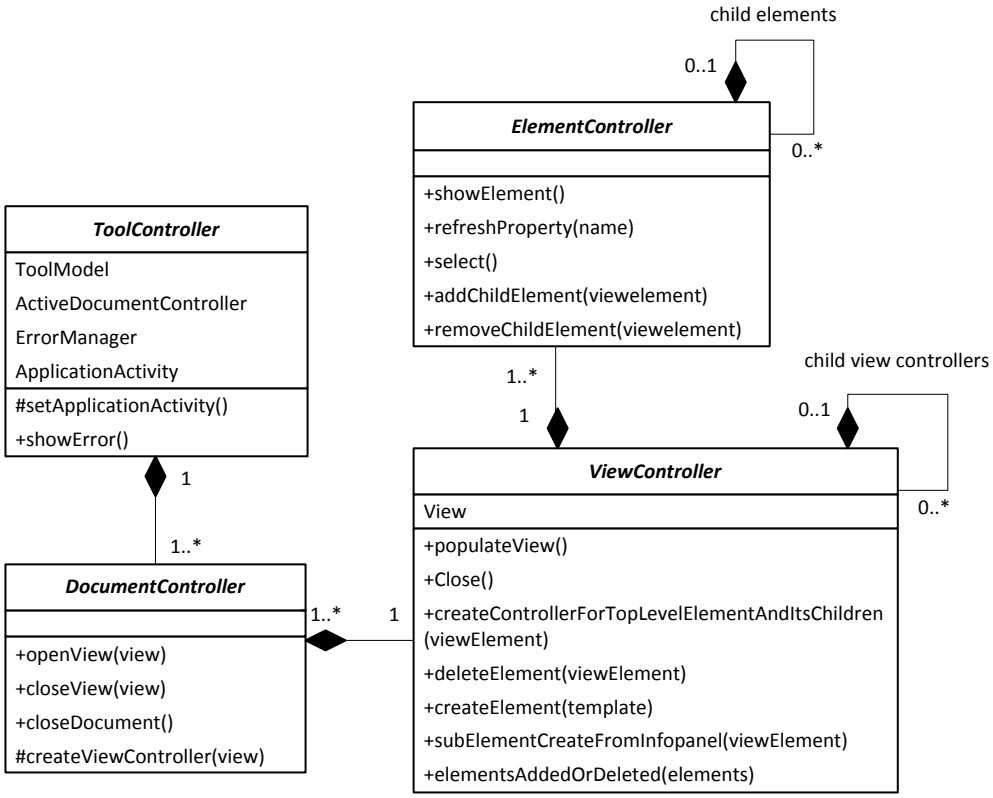
Agenttien välittämisestä alueiden ja paikkojen välillä huolehtii keskitetty komponentti, *AAFW* (Agent Architecture Framework), joka täytyy toteuttaa erikseen jokaiselle käytetylle ohjelmointikielelle. Trinityn agenttiarkkitehtuuri on toteutettu C#- ja Java-ohjelmointikielillä, joka mahdollistaa agenttien, alueiden, paikkojen ja tehtävien toteuttamisen niiden avulla. Tämän lisäksi olemassaolevia agentteja voi käynnistää myös PHP-ohjelmointikielellä. Sillä ei kuitenkaan voi toteuttaa agentteja, alueita, paikkoja tai tehtäviä, koska sille ei ole toteutettu agenttiarkkitehtuuria.

6.3 Tool-ohjelmistokehys

Tool-ohjelmistokehys tarjoaa pohjatoteutuksen uusille ympäristöön integroitaville Microsoft Office -pohjaisille mallinnustyökaluille, mukaanlukien Wordille. Se on modulaarinen ohjelmistokehys, joka koostuu joukosta abstrakteja kantaluokkia, joista sitä käyttävät työkalut periyttävät omat toteutuksensa. Luokat on jaoteltu moduuleihin niiden käyttötarkoituksen mukaan. Ohjelmistokehyksen määrittelemät moduulit ovat DBHandlers, Helpers, MainController, Model ja ViewController.

DBHandlers sisältää käsittelijöitä tietokannassa olevalle mallinnustyökalukohtaiselle informaatiolle. *Helpers* tarjoaa kaikkien ympäristön mallinnustyökalujen yleisesti tarvitsemia, mutta niiden sisäisistä toteutuksista riippumattomia sekalaisia aputoimintoja. *MainController* huolehtii kommunikaatiosta ympäristön integrointiarkkitehtuurin kanssa. Varsinaisissa toteutuksissa se huolehtii myös laajennoksen käynnistämisestä ja alustamisesta. *Model-moduulin* tehtävänä on huolehtia ympäristön tietomallista ja sen tarjoamisesta työkaluille. *ViewController* yhdistää *Model*-moduulin työkalun käyttöliittymään ja päivittää käyttöliittymän kautta tehdyt muutokset ympäristön tietomalliin ja toisin päin. Ote sen arkkitehtuurista on kuvattu kuvassa 6.2. Sen koko arkkitehtuuri on esitelty liitteessä 2.

Tool määrittelee sitä käyttävien työkalujen arkkitehtuurin korkealla tasolla. Siihen pohjautuvat mallinnustyökalut ovat MVC-arkkitehtuurimallin mukaisia. Se on suunniteltu siten, että arkkitehtuurimallin jokaista kerrosta vastaa joku komponentti valmiissa työkalussa. Mallikerrosta vastaa ohjelmistokehyksen *Model*-moduuli. Vastaavasti *ViewController*-moduuli vastaa arkkitehtuurimallin ohjainkerrosta. Laajennettava sovellusohjelma vastaa arkkitehtuurimallin näkymäkerrosta. Ohjelmistokehyksen *Helpers*-, *DBHandlers*- ja *MainController*-moduuleita ei voida sijoittaa millekään yksittäisille MVC-mallin kerroksille. Tool-ohjelmistokehystä käytetään kirjoitushetkellä tässä työssä kuvattavan Word-laajennoksen lisäksi ympäristöön kuuluvassa, Microsoft Excel -pohjaisessa mallinnustyökalussa [6].



Kuva 6.2: Ote Tool-ohjelmistokehyksen ViewController-kerroksen arkkitehtuurista.

6.4 Laajennoksen toteutustekniikat

Word-laajennos toteutettiin pääosin ohjelmatason add-in-laajennoksena, mutta osa ominaisuuksista toteutettiin käyttäen makrolaajennosta. Pääasialliseksi laajennostyypiksi valittiin add-in-laajennos, sillä niiden kehittämiseen on makrolaajennoksia paremmat työkalut, ja niistä on helpompaa integroitua muuhun ympäristöön. Laajennos päädyttiin toteuttamaan ohjelmatason laajennoksena, sillä tarkoituksena oli laajentaa koko Wordia, ei yksittäistä asiakirjaa.

Laajennoksen pääasialliseksi toteutusteknologiaksi valittiin Microsoft .NET Framework -ohjelmistokehys. Pääasialliseksi toteutusohjelmointikieleksi valittiin C#. Tähän ratkaisuun päädyttiin, sillä ne tarjoavat suoran tuen Microsoft Officen laajentamiseen, ja koska ympäristön muutkin palvelut ja komponentit oli toteutettu niiden avulla. Niitä myös käytetään yleisesti Officen laajentamiseen ja niistä on saatavilla paljon tietoa siihen liittyen, joka helpottaa ratkaisujen etsimistä ongelmatilanteissa. Makrolaajennoksen vaatimat osat toteutettiin Visual Basic for Applications -ohjelmointikielellä.

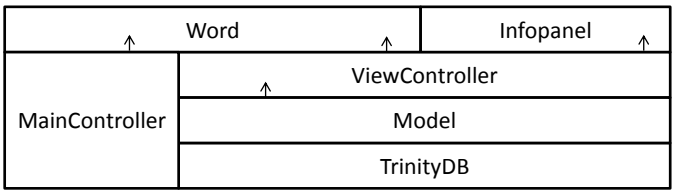
Työkalun kehitys aloitettiin Microsoft Wordin versiolla 2007, .NET Frameworkin versiolla 3.5 SP1 ja C#:n versiolla 3.5 Toteutuksen edistyessä toteutusteknologiat kuitenkin vaihtuivat Microsoft Word 2010:een, .NET Framework 4.0:aan sekä C# 4.0:aan. Nämä muutokset eivät vaikuttaneet työkalun arkkitehtuuriin merkittävästi.

6.5 Laajennoksen arkkitehtuuri

Word-laajennoksen arkkitehtuuri perustuu luvussa 6.3 esiteltyyn Tool-ohjelmistokehykseen, ja täten sen arkkitehtuuri perustuu MVC-malliin. Tämän lisäksi arkkitehtuuri voidaan ajatella kerrosarkkitehtuurina, sillä se on jaettu selkeisiin kerroksiin. Kukin kerros on tekemisissä ainoastaan välittömästi sen ylä- ja alapuolella olevien kerrosten kanssa, joten kerrosten välisiä ohituksia ei synny. Osasta laajennoksen kerroksista on pääsy myös niiden yläpuolisten kerrosten palveluihin, joten arkkitehtuuri ei ole puhtaan kerrosarkkitehtuurin mukainen. Laajennoksen kerrosarkkitehtuuri on kuvattu kuvassa 6.3.

Kerrokset kommunikoivat toistensa kanssa tapahtumien ja funktiokutsujen avulla. Yleisenä periaatteena on, että kerroksen sisäistä tilaa voidaan muuttaa funktiokutsujen avulla. Näistä muutoksista ilmoitetaan muille kerroksille tapahtumien avulla. Tapahtumat ja niiden käsittelijät perustuvat Tarkkailija-suunnittelumalliin ja ne on toteutettu .NET Frameworkin tapahtumamekanismin avulla.

Tool-ohjelmistokehyksen arkkitehtuurin mukaisesti laajennoksen näkymäkerroksena (View) toimivat sekä itse Word että ympäristön informaatiopaneeli. Täten sen rajapinta koostuu Wordin automaatiorajapinnasta ja informaatiopaneelin ulkoisesta rajapinnasta.

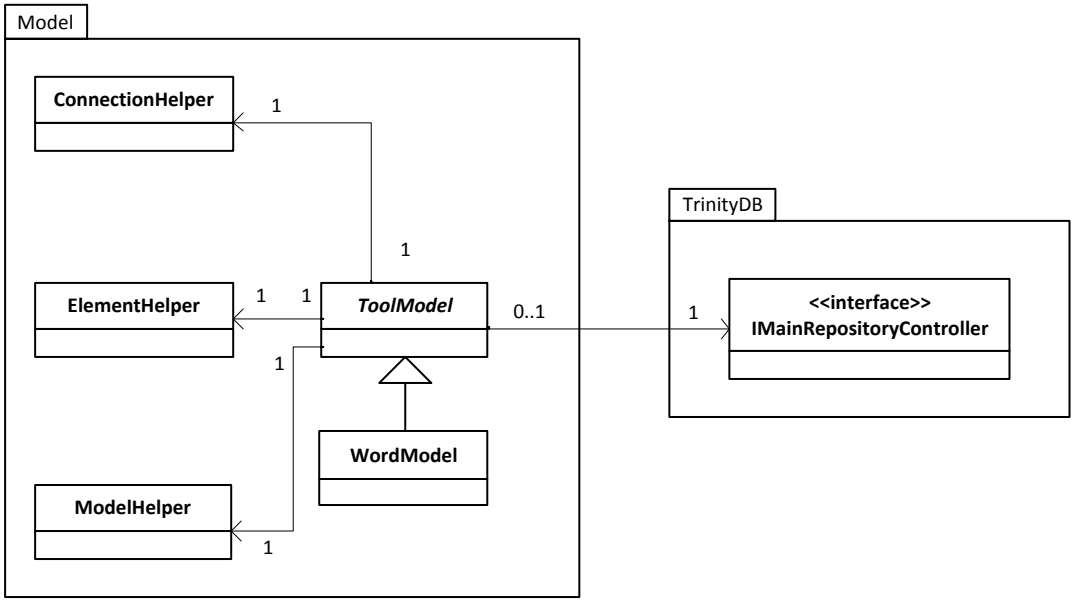


Kuva 6.3: Laajennoksen kerrosarkkitehtuuri. Nuolet kuvaavat sitä, että kerros, josta nuoli lähtee, käyttää ylemmän kerroksen palveluita.

6.5.1 Model-kerros

Laajennoksen mallikerros (Model) koostuu kahdesta moduulista: Tool-ohjelmistokehyksen mukaisesta *Model*-moduulista ja ympäristön *tietokantakomponentista*, joka esiteltiin luvussa 6.2.2. Sen ensisijainen tehtävä on ylläpitää ympäristön tietomallia ja hallinnoida siihen kohdistuvia muutoksia. Se välittää tiedon siihen kohdistuvista ulkoisista muutoksista ohjainkerrokselle ja tarjoaa sille mekanismit sen muuttamiseen. Tämän lisäksi se tarjoaa kommunikointikanavan agenttiarkkitehtuurin ja ohjainkerroksen välille. Model-moduulin arkkitehtuuri on esitetty kuvassa 6.4.

Model-moduuli koostuu Tool-ohjelmistokehyksen tarjoamasta pohjatoteutuksesta sekä WordModel-luokasta, joka on periytetty ohjelmistokehyksen *ToolModel-luokasta*. *WordModel-luokka* laajentaa ToolModel-luokkaa yhdistämällä sen Word-toteutuksen DBHandlers-moduuliin. Ohjelmistokehyksen tarjoama Model-kerroksen

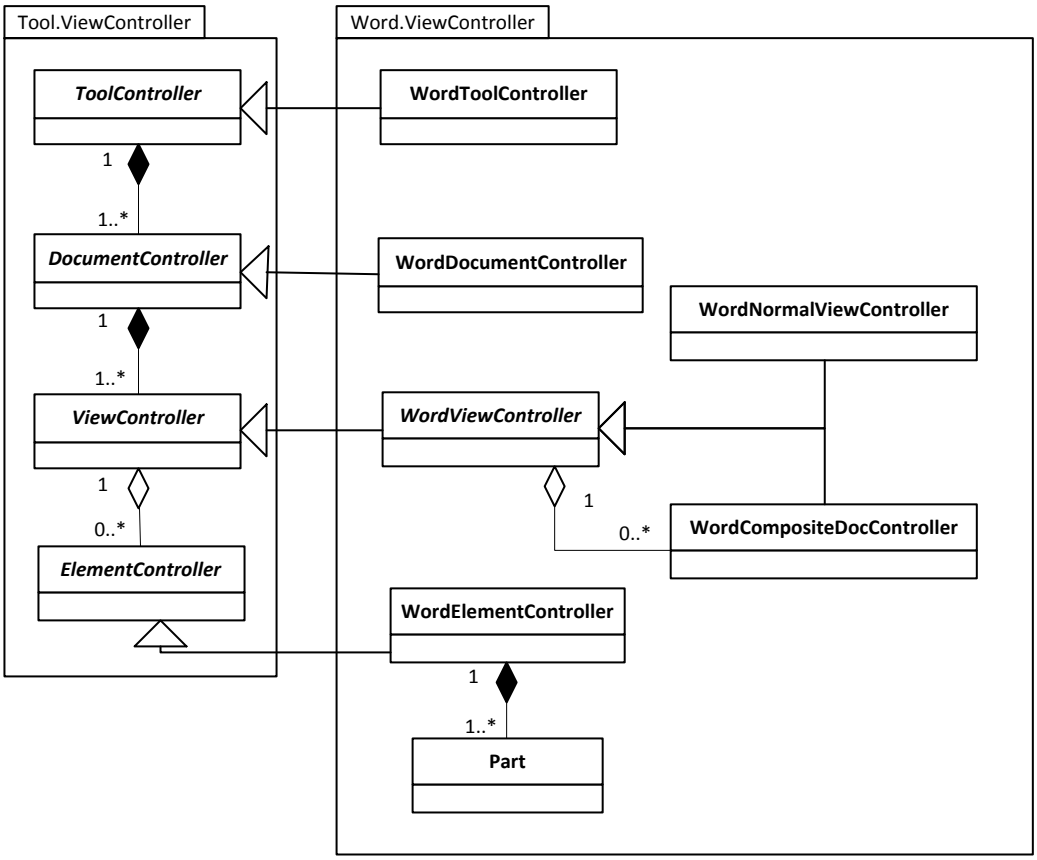


Kuva 6.4: Laajennoksen Model-moduuli.

tarkoituksena on tarjota apufunktioita tietokannan sisällön käsittelyyn, kuten esimerkiksi näkymien avaamiseen ja uusien näkymäelementtien luomiseen. Pohjatoteutus on jaettu neljään luokkaan, jotka ovat ToolModel, *ElementHelper*, *ConnectionHelper* ja *ModelHelper*. ToolModel ei säilytä tietoa ympäristön tietomallin tilasta, vaan se on tietokantakomponentin vastuulla.

ToolModel on kerroksen keskeisin luokka, ja se tarjoaa pääsyn sekä moduulin muihin palveluihin että ympäristön tietokantakomponenttiin. Sen julkinen rajapinta toimii myös koko Model-kerroksen rajapintana ViewController-kerroksen suuntaan. Se tarjoaa myös itse yleisiä palveluita, kuten kerroksen alustustoimenpiteet sekä näkymien avaamis- ja sulkemistoimenpiteet. Tämän lisäksi se mahdollistaa ViewController-kerroksen kommunikoinnin agenttiarkkitehtuurin kanssa tarjoamalla sille pääsyn Word-laajennoksen paikkaan.

ElementHelper-luokka sisältää apufunktiota helpottamaan malli- ja näkymäelementtien käsittelyä. Sen vastuulla on muun muassa uusien malli- ja näkymäelementtien luominen. Tämän lisäksi se tarjoaa palveluita kuten elementtihierarkian ylimmän tason elementin etsimisen ja tarkastelun siitä, minkä tyyppisiä lapsielementtejä jollain elementillä voi olla. **ConnectionHelper** sisältää apufunktioita näkymäelementtien välisten suhteiden hallintaan, kuten lapsielementtien liittämisen isäelementtiin ja elementtien välisen suhteen katkaisemisen. **ModelHelper** sisältää apufunktioita ympäristön mallien hallintaan.



Kuva 6.5: Laajennoksen ViewController-moduulin keskeisimmät luokat.

6.5.2 ViewController-kerros

ViewController-kerros toimii MVC-mallin mukaisena ohjainkerroksena (Controller). Sen tarkoitus on toimia siltana Model- ja View-kerrosten välillä ja pitää niiden tilat synkronoituna toisiinsa. Se kuuntelee Model- ja View-kerrosten laukaisemia tapahtumia tilan muutoksesta ja päivittää vastakkaisen kerroksen tilan kutsumalla sen funktioita.

ViewController-kerroksen keskeisimmät luokat on kuvattu kuvassa 6.5. Ne ovat *WordToolController*, *WordDocumentController*, *WordViewController* ja sen lapsiluokat, *WordElementController* ja *Part*. Luokat, joiden nimissä on Word-etuliite, on nimetty siten, jotta ne erottuisivat Tool-ohjelmistokehyksen ViewController-moduulin samannimisistä luokista. Part-luokkaa lukuunottamatta kaikki edellämainitut luokat periytyvät niiden Tool-ohjelmistokehyksessä olevista vastineista.

WordToolControllerin vastuulla on WordDocumentController-luokan olioiden organisointi. Se kuuntelee agenttiarkkitehtuurilta tulevia kutsuja, kuten esimerkiksi käskyjä avata jokin näkymä ja välittää ne eteenpäin sopivalle WordDocumentControllerille. WordToolController huolehtii myös WordDocumentController-olioiden luomisesta ja tuhoamisesta. Esimerkiksi ensimmäistä näkymää ladattaessa

se luo uuden WordDocumentController-olion, jonka se käskee avaamaan avattavan näkymän. Tämän lisäksi se huolehtii käynnissä olevan Word-instanssin sellaisten toimintojen hallinnasta, jotka vaikuttavat kaikkiin auki oleviin asiakirjoihin. Esimerkki tästä on koko Wordin sulkeminen, jolloin WordToolController ottaa kiinni siitä aiheutuvan tapahtuman ja sulkee automaattisesti kaikki auki olevat näkymät.

WordDocumentController-luokan tehtävä on yksittäisen Word-asiakirjan ja siihen avattujen näkymien hallinnointi. Jokaiselle Word-asiakirjalle ei luoda omaa WordDocumentController-olioa, vain ainoastaan niille, joihin on avattu vähintään yksi Trinity-näkymä. Se kuuntelee Word-asiakirjan laukaisemia tapahtumia ja toimii niiden perusteella. Tärkein luokan kuuntelema tapahtuma on *SelectionChange*, joka kuvaa käyttäjän käsin tekemää muutosta tekstinsyöttökursorin sijaintiin. Se toimii pohjana muutosten tallennusmekanismille, sillä asiakirjalta ei ole mahdollista saada suoremmin tietoa käyttäjän asiakirjaan tekemistä muutoksista.

WordViewController-luokan ja sen lapsiluokkien tehtävänä on huolehtia yksittäisestä näkymästä ja sen sisältämistä näkymäelementeistä. Itse WordViewController-luokka on abstrakti kantaluokka, joka tarjoaa palveluita, jotka ovat yhteisiä kaikille siitä periytyville lapsiluokille. Siitä on periytetty luokat WordNormalViewController, joka kuvaa yksittäistä näkymää, ja WordCompositeDocViewController, joka kuvaa näkymähierarkian osana olevaa näkymää. WordDocumentController luo jokaiselle avatulle näkymälle sitä vastaavan WordViewController-olion. Luomisoperaatio hoidetaan Tehdasmetodi- ja Abstrakti tehdas -suunnittelumallien yhdistelmänä, joten WordDocumentControllerin ei tarvitse tietää kumpaa tyyppiä oleva olio kullekin näkymälle luodaan.

WordViewController-luokka hallitsee näkymäänsä liittyviä näkymäelementtejä huolehtimalla niiden luomisesta, poistamisesta ja asemoinnista. Esimerkiksi luotaessa uutta elementtiä valintanauhan Trinity-välilehden kautta WordDocumentController välittää tiedon kyseisellä hetkellä valittuna olevalle WordViewControllerille, joka etsii sille sopivan paikan näkymästä ja luo siihen halutun tyyppisen elementin. Tämän lisäksi se tarkistaa mitä elementtejä WordDocumentControllerilta tulevat SelectionChange-tapahtumat koskevat ja pyytää niitä tallentamaan itsensä tietokantaan mikäli ne ovat muuttuneet.

WordViewController luo jokaiselle näkymän näkymäelementille oman **WordElementController-luokan** olion, joka kuvaa kyseistä näkymäelementtiä. Se yhdistää tietomallin näkymäelementtikuvauksen joukoksi *Part-luokan olioita* ja pitää ne synkronoituna. Se saa tiedon muiden käyttäjien tietomalliin tekemistä muutoksista tietokantakomponentin *IViewElement-* ja *IModelElement-rajapintojen* tapahtumien kautta. Se myös kuuntelee WordViewControllerin sille välittämiä SelectionChange-tapahtumia. SelectionChange-tapahtuman saapuessa se tarkistaa, onko elementti muuttunut ja jos on, se tallentaa itsensä tietokantaan. WordElementController huo-

maa myös, jos jokin elementti poistetaan näkymästä. Tällöin se merkitsee sen poistetuksi tietokannassa ja ilmoittaa siitä sille WordViewControllerille, josta kyseinen näkymäelementti löytyy.

WordElementController-luokan oliot koostuvat **Part-olioista**. Jokainen Part-olio kuvaa yhden tietokannasta löytyvän näkymäelementin osan Wordin *Bookmark-oliona* ja pitää ne synkronoituna. Esimerkiksi kun tieto SelectionChange-tapahtumasta saapuu WordElementControllerille, se kysyy kaikilta sen osilta ovatko ne muuttuneet tai poistettu. Yksittäinen Part huomaa, jos sen kuvaava Bookmark tuhotaan näkymästä. Tällöin kyseinen Part ilmoittaa WordElementControllerille, että se on tuhoutunut, ja sekä se että sitä vastaava osa voidaan poistaa elementistä. Part osaa myös tarvittaessa jakaa itsensä uusiin Part-olioihin. Tätä toiminnallisuutta tarvitaan esimerkiksi käyttäjän vaihtaessa jotain tyyli- tai typografia-asetusta olemassaolevan Partin sisällä.

6.6 Word-laajennoksen erityisominaisuudet

Tässä luvussa on kuvattu tärkeimpien sellaisten Word-laajennoksen ominaisuuksien toteutus, jotka liittyvät laajennoksen erityispiirteisiin. Näitä ovat näkymähierarkioiden esittäminen asiakirjamaisena rakenteena, Visio- ja Excel-näkymien näyttäminen osana näkymähierarkioita, raporttien luominen ja asiakirjojen muuttaminen mallidataksi.

6.6.1 Visio- ja Excel-näkymien näyttäminen Wordissa

Jokaiselle Wordissa näytettävälle Visio- ja Excel-näkymälle luodaan *WordCompositeDocViewController-luokan* olio. Word-laajennos ei itsessään osaa tulkita niille tarkoitettua näkymädataa, joten näkymien tulkitseminen tehdään siinä työkalussa, jolla niitä on tarkoitettu muokattavan. Näkymien sisältö tallennetaan kohdetyökaluissa väliaikaistiedostoihin, josta Word-laajennos lukee niiden sisällön osaksi asiakirjaa. Kaikki väliaikaistiedostot poistetaan työkalun sulkeutuessa.

Kommunikaatio työkalujen välillä hoidetaan agenttien avulla. Kun näkymähierarkiassa on tarve avata jollekin muulle työkalulle tarkoitettu näkymä, Word-laajennos lähettää kohdetyökalulle *SaveViewAgent-agentin*. Sen tehtävänä on tallentaa näkymä väliaikaistiedostoon. Jos kohdetyökalu ei ole päällä, agentti avaa sen. Tämän jälkeen agentti pyytää työkalua tallentamaan näkymän väliaikaistiedostoon. Kun näkymä on tallennettu, agentti palaa takaisin Word-laajennokseen ilmoittamaan tallennusoperaation loppumisesta. Näkymän päivittäminen malli- ja näkymädatan muuttuessa hoidetaan samalla mekanismilla: kun tietokantakomponentti ilmoittaa muutoksesta, Word-laajennos lähettää kohdetyökalulle uuden SaveViewAgent-agentin, joka tallentaa päivittyneen näkymän väliaikaistiedostoon. Agentin palatessa Word-

laajennokselle kyseisen näkymän edellistä tilaa kuvaava kuva tai taulukko korvataan uudella vastaavalla.

Visio-näkymät tallennetaan vektorigrafiikkana Extended Metafile (.emf) -tiedostomuodossa. Ne sisällytetään näkymähierarkiaa kuvaavaan asiakirjaan Wordin automaatiorajapinnan *InlineShape-olioina*. Näkymät esitetään kuvina muiden esitysmuotojen, kuten esimerkiksi OLE-objektien sijasta, sillä kuvien käsittely on tehokasta.

Excel-näkymät tallennetaan Excel 2010 -ohjelmalle luontaisessa Microsoft Office Open XML Spreadsheet (.xlsx) -tiedostoformaatissa, joita myös Word tukee. Ne sisällytetään näkymähierarkiaan Wordin normaalilla taulukkomekanismilla. Sitä käytetään kehittyneempien esitysmuotojen sijasta, sillä useimmilla muilla esitysmuodoilla, kuten Wordin ja Excelin välisellä taulukkolinkityksellä tai OLE-objekteina taulukot voivat viedä maksimissaan yhden sivun. Ne taulukon osat, jotka eivät mahdu sivulle, jouduttaisiin tällöin jättämään kokonaan näyttämättä. Excel-näkymät voivat kuitenkin olla paljon pidempiä, joten ne täytyy esittää mekanismilla, joka tukee monelle sivulle jakautuvia taulukoita.

6.6.2 Näkymähierarkiat

Näkymähierarkioiden muodostamat puumaiset rakenteet kuvataan WordViewController-luokan olioina Kooste-suunnittelumallin mukaisesti. Jokaiselle näkymähierarkian näkymälle luodaan oma WordViewController-luokan olio. Kukin näkymähierarkian osana oleva näkymä avataan muuten samalla tavalla kuin muutkin näkymät, mutta avaamisprosessin lopussa tarkistetaan onko avatulla näkymällä lapsia. Lapsinäkymät käydään läpi ja avataan asiakirjaan syvyyssuuntaista läpikäyntiä hyväksikäyttäen. Tällä tavalla jokainen läpikäytävä näkymä voidaan avata heti edellisenä avatun perään.

Näkymähierarkiat toteutetaan Trinity-ympäristössä näkymien välisten suhteiden avulla ja ympäristö sallii tilanteen, jossa näkymähierarkia muodostaa silmukkamaisen rakenteen. Esimerkki tästä on tilanne, jossa jokin näkymä on yhdistetty lapsinäkymäksi jollekin sen lapsinäkymälle. Tämä on otettu huomioon laajennoksessa siten, että näkymää avattaessa tarkistetaan onko se jo avattu johonkin asiakirjaan. Jos se on jo valmiiksi auki, sitä tai sen lapsinäkymiä ei avata osana näkymähierarkiaa. Tämä tarkoittaa myös sitä, että tietyissä erikoistilanteissa koko näkymähierarkiaa ei avata asiakirjaan. Esimerkki tällaisesta tilanteesta on se, jossa jokin näkymähierarkiaan kuulumaton näkymä avataan sellaisenaan laajennokseen, yhdistetään osaksi näkymähierarkiaa tämän jälkeen, ja kyseinen näkymähierarkia avataan laajennoksessa.

6.6.3 Raporttien luominen

Raportin luominen laukaistaan ulkoisesta mallinnustyökalusta, kuten web-käyttöliittymästä. Tämä tehdään lähettämällä Word-laajennokselle agentti, jolle annetaan parametriksi sen mallielementin tunnistamiseen vaadittavat tiedot, jonka pohjalta raportti tehdään. Käytettävä raporttipohja valitaan automaattisesti mallielementin tyypin perusteella. Raporttipohja kopioidaan sellaisenaan uuteen Word-asiakirjaan, jonka jälkeen se käydään läpi ohjelmallisesti alusta loppuun. Tällöin raportissa merkityt mallielementtien ominaisuudet korvataan niiden oikeilla arvoilla. Valmis raportti tallennetaan väliaikaistiedostoksi ja tallennetaan tavuina tietokantaan uudeksi mallielementiksi. Lopuksi uusi, raporttia kuvaava mallielementti yhdistetään alkuperäiseen dokumentin ja mallielementin välistä yhteyttä kuvaavalla suhteella.

6.6.4 Asiakirjojen muuttaminen mallidataksi

Asiakirjojen muuttaminen mallidataksi aloitetaan pilkkomalla haluttu asiakirja osa-asiakirjoihin (subdocument) jokaisen asiakirjan otsikon kohdalta. Kukin osa-asiakirja tallennetaan väliaikaistiedostoon. Tämän jälkeen käyttäjän valitsemaan malliin luodaan mallielementit sekä koko kuvattavalle asiakirjalle että jokaiselle osa-asiakirjalle. Mallielementit yhdistetään toisiinsa suhteilla hierarkiaksi, joka vastaa alkuperäisen asiakirjan rakennetta. Lopuksi kukin väliaikaistiedosto tallennetaan sitä kuvaavaan mallielementtiin.

Muunnettavat asiakirjat tallennetaan tietokantaan samassa muodossa kuin raportit, eli Microsoft Office Open XML Document (.docx) -muotoisina tiedostoina. Tämä johtuu siitä, että muunnettavien asiakirjojen sisältöä ei ole mahdollista analysoida koneellisesti niin tarkasti, että siitä voitaisiin löytää malleja, näkymiä tai mallielementtejä. Esimerkiksi muutettaessa olemassaolevaa suunnitteludokumenttia mallidataksi ympäristöön on erittäin hankalaa tunnistaa, missä esimerkiksi luokkien kuvaukset ovat, tai vaikka tämä saataisiinkin tehtyä, missä luokkien ominaisuudet, kuten niiden nimet, on kuvattu.

7. TYÖN ARVIOINTI

Tässä luvussa esitetään luvuissa 3.3 ja 4.3 esitettyjen vaatimusten toteutumisen arviointi. Tämän lisäksi käydään läpi Trinity Word Add-inin tulevaisuutta ja parannusehdotuksia. Lopuksi esitellään muita julkaisuja, jotka liittyvät tämän työn aiheeseen.

7.1 Vaatimusten toteutumisen arviointi

Vaatimusten toteutumista arvioidaan vaatimuskattavuusmatriisilla. Arviointi suoritetaan käyttämällä kolmiportaista asteikkoa, joka on kuvattu taulukossa 7.1. Kaikki laajennokselle asetetut vaatimukset saatiin täytettyä, joten asteikossa ei ole arvoa, joka kuvaisi toteutumatonta vaatimusta.

Taulukko 7.1: Vaatimusten toteutumisen arvioinnissa käytetty asteikko

Arvo	**Kuvaus**
+	Vaatimuksen täyttämisessä onnistuttiin hyvin.
0	Vaatimuksen täyttämisessä onnistuttiin kohtuullisesti.
-	Vaatimuksen täyttämisessä onnistuttiin heikosti.

Taulukossa 7.2 on kuvattu Trinity Word Add-inille asetetut vaatimukset ja niiden arviointi. Vaatimukset on ryhmitelty kategorioittain.

Käyttöliittymälle asetetut vaatimukset onnistuttiin täyttämään pääasiassa hyvin. Laajennoksen avulla on tämän työn kirjoitushetkellä mahdollista mallintaa käyttötapauksia. Laajennoksen avulla mallinnettu esimerkkikäyttötapaus löytyy liitteestä 3. Mallinnusominaisuuksien integroiminen Wordiin sen sisäisillä mekanismeilla mahdollistaa sen, että mallidatan luominen ja muokkaaminen on mahdollista tekstinkäsittelyohjelmille tyypillisillä työskentelytavoilla, kuten puhdasta tekstiä syöttämällä. Elementtejä luotaessa elementtipohjien avulla laajennoksen suorituskyky on kuitenkin huono, sillä Wordin rajapinnan dokumenttia muokkaavat operaatiot ovat hitaita. Tästä johtuen myös näkymien avaaminen on hidasta.

Asiakirjojen mallidataksi muuntamisen vaatimukset saatiin täytettyä pääosin hyvin, mutta kokonaisuutena toiminnallisuudessa on silti joitain puutteita. Muunnosprosessin käynnistys tapahtuu tällä hetkellä Wordista, vaikka parempi paikka sille voisi olla esimerkiksi ympäristön hallintakäyttöliittymä. Samoin asiakirjojen

Taulukko 7.2: Mallinnustyökalun vaatimusten arviointi

Tunniste ja nimi	**Arvio**
Käyttöliittymä	
M01, Työskentelytapa	+
M02, Elementtien muokkaaminen	+
M03, Elementtien luominen pohjien avulla	0
M04, Elementtien luominen vapaamuotoisesti	+
M05, Mallielementtien ulkopuolinen näkymädata	+
Asiakirjojen muuttaminen mallidataksi	
M06, Asiakirjojen muuntaminen mallidataksi	0
M07, Muunnettujen asiakirjojen käsitteleminen	+
M08, Asiakirjojen analysointi	+
M09, Muunnosprosessin automatisointi	+
Raportointi	
M10, Raporttien luominen	+
M11, Raporttien tallentaminen	+
M12, Raporttien yhdistäminen elementteihin	+
M13, Raporttien luonnin etäkäynnistäminen	+
M14, Raporttien luominen palvelimella	0
Ylläpito	
M15, Elementtipohjien ylläpito	0
M16, Raporttipohjien ylläpito	-
Joustava mallintaminen	
T01, Mallinnusominaisuuksien kytkeytyminen päälle	+
T02, Perustoiminnallisuuksien säilyttäminen	+
T03, Työskentelytapojen tukeminen	+
T04, Käyttöliittymän säilyttäminen samanlaisena	+
T05, Rikkinäiset mallit	+
Tietomalli ja samanaikainen työskentely	
T06, Tietomalli	+
T07, Muutosten tallentaminen	0
T08, Muutosten hakeminen	0
Ympäristön työkalulle asettamat muut vaatimukset	
T09, Väritykset	+
T10, Näkymähierarkioiden esittäminen	+
T11, Muille työkaluille määriteltyjen näkymien esittäminen	0

sisältämä tieto tallennetaan tietokantaan .docx-tiedostoja sisältävinä mallielementteinä. Tämä on ristiriidassa ympäristön näkymäelementtimekanismin kanssa.

Raporttien luominen toimii käyttäjän näkökulmasta suurimmaksi osaksi hyvin, mutta kokonaisuudessa on silti joitain puutteita. Raportin luominen on mahdollista suorittaa esimerkiksi palvelimella, mutta toiminnallisuuden käyttöä rajoittaa se, että Wordia ei ole suunniteltu käytettäväksi täysin autonomisesti: Word saattaa esimerkiksi avata käynnistyessään jonkun dialogi-ikkunan, jonka päälläolon aikana teknisistä rajoitteista johtuen ei ole mahdollista käyttää Wordin oliomallin rajapintoja ja jota ei ole mahdollista sulkea ohjelmallisesti. Raportin luominen on melko hidas operaatio: jo muutaman sivun mittaisen raportin luomisessa saattaa kestää yli minuutti. Raportit tallennetaan tietokantaan .docx-tiedostoina, joka on ristiriidassa ympäristön näkymäelementtimekanismin kanssa. Se ei myöskään mahdollista mallidatan muokkaamista raportin kautta.

Laajennoksen ylläpitovaatimukset saatiin täytettyä kohtuullisesti. Elementtipohjien ylläpito on mahdollista laajennoksen avulla, ja käyttötapausnäkymätyypin elementtipohjat on luotu tämän toiminnon avulla. Elementtipohjien ylläpito on kuitenkin hankalaa: syntaksin oikeellisuuden tarkastaminen ei esimerkiksi ole mahdollista ennen pohjan tallentamista, eikä missään ole lueteltu pohjalle annettavia mahdollisia metaominaisuuksia tai mallielementtityyppiin liittyviä ominaisuuksia ja niiden tietotyyppejä. Sama kritiikki voidaan osoittaa myös raporttipohjien hallinnointiin. Tämän lisäksi raporttipohjat tallennetaan ympäristön asennushakemistoon, eikä esimerkiksi tietokantaan, mikä on ristiriidassa ympäristön perusperiaatteiden kanssa.

Trinity-ympäristön laajennokselle asettamat **joustavan mallintamisen vaatimukset** saatiin toteutettua hyvin. Laajennokselle asetetut **ympäristön tietomallin tukemisen ja samanaikaisen työskentelyn vaatimukset** saatiin täytettyä, mutta muutosten tallentamisessa ja hakemisessa on Wordin rajapinnan teknisistä rajoitteista johtuvia ja laajennoksen käyttökokemusta huonontavia puutteita. Muutosten tallentaminen jouduttiin toteuttamaan käyttäjän vaihtaessa Wordin tekstinsyöttökursorin paikkaa eksplisiittisesti. Täten on mahdollista, että käyttäjien tekemät muutokset päivittyvät tietokantaan epätasaisesti. Muiden näkymään tekemien muutosten päivittäminen Wordiin on myös hidasta, sillä joissain tapauksissa koko muuttunut näkymäelementti joudutaan rakentamaan uudestaan.

Muut ympäristön työkalulle asettamat vaatimukset saatiin täytettyä hyvin, joskin muille työkaluille tarkoitettujen näkymien esittämisessä on vielä pieniä puutteita. Laajennoksessa näytetyt Excel-näkymät eivät päivity automaattisesti, kun niiden sisältämä mallidata muuttuu. Niihin tehdyt muutokset eivät myöskään tallennu tietokantaan, vaikka ne esitetään dokumentissa muokattavissa olevana taulukkona.

7.2 Trinity Word Add-inin tulevaisuus ja parannusehdotuksia

Laajennoksen toteutus ei ole vielä valmis, ja sitä tullaan jatkokehittämään tulevaisuudessa. Osa ominaisuuksista, kuten raportointimekanismi ja asiakirjojen muuntaminen mallidataksi, ovat vielä keskeneräisiä ja siksi ristiriidassa ympäristön perusperiaatteiden kanssa. Myös laajennoksen **suorituskyvyssä** on kehitettävää, sillä esimerkiksi näkymien avaaminen, uusien elementtien luonti mallipohjien perusteella ja muiden käyttäjien tekemien muutosten päivittäminen näkymään eivät ole tällä hetkellä riittävän suorituskykyisiä. Laajennos tukee tällä hetkellä vain käyttötapausten mallintamista, ja laajennoksen tukemia mallinnuskieliä tullaan lisäämään tulevaisuudessa.

Laajennoksen **arkkitehtuurissa** on ylläpidettävyyden kannalta parannettavaa elementtien ja niiden osien käsittelyssä. Laajennoksen nykyisessä versiossa erilaisten elementtityyppien (esim. korkeimman tason elementit, lokeroelementit, lapsielementit) käsittely on keskitetty ViewController-moduulin WordElementController-luokalle. Sen olisi järkevää hajottaa luokkahierarkiaksi, jossa jokainen luokka kuvaa eri tyyppistä elementtiä. Näkymäelementtien osat on toteutettu tällä hetkellä Part-luokkana, joka huolehtii sekä puhtaan tekstin, mallielementtien ominaisuuksien että lapsielementtiosien esittämisestä. Myös Part-luokka olisi kannattavaa hajottaa luokkahierarkiaksi, jossa jokainen luokka kuvaa erilaista näkymäelementin osaa.

Laajennos ei ole kirjoitushetkellä ollut käytössä oikeilla käyttäjillä, vaan sitä on käytetty ainoastaan Trinityn kehitysorganisaation sisäisesti. Täten laajennoksen käyttäminen oikeassa mallinnustyössä paljastanee sen ominaisuuksista ja toiminnasta lisää asioita, jotka toimivat huonosti.

7.3 Aiheeseen liittyviä julkaisuja

Xia ja muut esittelevät julkaisussaan [23] laajennoksen Microsoft Wordiin nimeltä *CoWord*. Sen avulla monta käyttäjää voi muokata samaa asiakirjaa reaaliaikaisesti. Se sivuaa tältä osin Trinity Word Add-iniä. CoWordin ratkaisumalli eroaa kuitenkin Trinityn ratkaisumallista siten, että Trinityssä käsiteltävä tieto on aina mallitai näkymädataa ja tallennetaan tietokantaan, kun CoWordissä käsitellään tiedostomuotoisia Word-dokumentteja. CoWord sisältää välikerroksen, joka kaappaa käyttäjän dokumenttiin kohdistamat syötteet ja muodostaa niiden perusteella komentoja, jotka välitetään verkon yli muille muokkaajille. Word ei itse tarjoa tapahtumia monista dokumenttiin kohdistetuista muutoksista, jonka takia päivitysten epätasaisuus on nykyisellään ongelma Trinity Word Add-inissä. CoWord tarjoaa yhden varteenotettavan ratkaisutavan tämän ongelman ratkaisemiseen.

Sybase PowerDesigner tarjoaa tavan vaatimusten mallintamiseen Word-dokumenttien avulla [21]. Vaatimuksia voidaan kuvata missä tahansa Word-dokumentissa

taulukoiden ja otsikoiden avulla. Dokumentteja on mahdollista yhdistää PowerDesignerissa olevaan malliin, jolloin malli päivitetään automaattisesti, kun dokumenttia muutetaan ja toisin päin. Tämä vastaa osittain Trinity Word Add-inin mallinnusominaisuuksia ja asiakirjojen muuntamista mallidataksi. Trinityn tarjoama Word-pohjainen mallinnustuki on tosin yleiskäyttöisempi kuin PowerDesignerin, sillä Trinity Word Add-inia voidaan laajentaa tukemaan uusia mallinnuskieliä.

PowerDesignerin vaatimustenmallinnusmekanismin avulla on mahdollista muuntaa asiakirjoja mallidataksi tarkemmin kuin Trinity Word Add-inin vastaavalla mekanismilla. Mallidatan tarkempi muodostaminen asiakirjojen pohjalta on yksi Word-laajennoksen mahdollisista tulevista toiminnallisuuksista. PowerDesignerin avulla on myös mahdollista luoda raportteja mallidatan perusteella [19]. Sen raportointimekanismi on Trinity Word Add-inin mekanismia monipuolisempi, ja edustaa kehityssuuntaa, johon Trinity Word Add-inia voidaan kehittää.

8. YHTEENVETO

Tässä työssä toteutettiin Microsoft Word -laajennos, Trinity Word Add-in, joka laajentaa Wordin ohjelmistojen mallinnustyökaluksi Trinity-mallinnus- ja työkaluympäristöön. Laajennos mahdollistaa sekä olemassaolevan mallidatan esittämisen Wordissa dokumenttimuotoisesti että mallidatan syöttämisen ja muokkaamisen Wordille ominaisten työskentelytapojen avulla. Mallidata tallennetaan ympäristön keskitettyyn tietokantaan, ja laajennos tukee montaa yhtäaikaista käyttäjää pitämällä tietokannan ja Wordissa näytetyn informaation synkronoituna. Se tarjoaa myös mahdollisuuden mallinnustuen laajentamiseen lisäämällä tukea uusille mallinnuskielille. Tämän lisäksi laajennos mahdollistaa raporttien luomisen mallidatan pohjalta sekä olemassaolevien asiakirjojen muuttamisen mallidataksi ympäristöön.

Kaikki laajennokselle asetetut vaatimukset saatiin täytettyä. Laajennoksen toteutuksesta saatiin uudelleenkäytettävä ja kohtuullisen geneerinen. Toisaalta laajennoksen suorituskyky ei ole vielä riittävän hyvällä tasolla, ja laajennoksen arkkitehtuuri on osittain liian monimutkainen. Tulevaisuudessa osa laajennoksen luokista tulisi hajottaa luokkahierarkioiksi, joissa jokaisella luokalla on ainoastaan yksi, selkeästi rajattu rooli. Myös Wordista johtuvia suorituskykyongelmia ja teknisiä rajoitteita tulisi pyrkiä kiertämään. Näiden lisäksi myös raportointiominaisuutta ja asiakirjojen muuntamisominaisuutta on pyrittävä parantamaan esimerkiksi siten, että niissä käytetty tiedon tallennustapa tukee paremmin ympäristön perusperiaatteita. Laajennoksen mallinnus-, raportointi- ja asiakirjojen muuntamisominaisuudet kuitenkin toimivat jo nyt, ja tarjoavat pohjan niiden jatkokehitykselle.

Työssä tutustuttiin myös aihetta sivuaviin julkaisuihin ja tuotteisiin. Yhtään täysin vastaavaa tuotetta ei löytynyt, mutta samoja ominaisuuksia sisältäviä tuotteita löytyi kaksi. CoWordin avulla monta käyttäjää voi muokata samaa Worddokumenttia samanaikaisesti. Sybase PowerDesigner mahdollistaa vaatimusten mallintamisen Microsoft Wordin avulla ja dokumenttimuotoisten raporttien luomisen mallidatan perusteella. Sen raportointiominaisuudet ovat Trinity Word Add-inin vastaavia monipuolisemmat. Trinity Word Add-in on kuitenkin mallinnuskäytössä PowerDesignerin Word-integraatiota yleiskäyttöisempi, sillä sitä on mahdollista laajentaa tukemaan uusia mallinnuskieliä.

LÄHDELUETTELO

[1] Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, Internet Engineering Task Force 2005, [WWW]. [Viitattu 9.5.2012]. http://tools.ietf.org/html/rfc3986

[2] Felin, M. Microsoft Vision laajentaminen joustavaksi tietokantapohjaiseksi mallinnustyökaluksi. Tampere 2011. Tampereen teknillinen yliopisto. Diplomityö. 60 s.

[3] Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design patterns: elements of reusable object-oriented software. 1995, Addison-Wesley Professional. 395 s.

[4] Haikala, I., Märijärvi, J. Ohjelmistotuotanto. Helsinki 2006, Talentum Media Oy. 440 s.

[5] Halme, K. Mallinnustyökaluympäristön hallintatyökalu. Tampere 2011. Tampereen teknillinen yliopisto. Diplomityö. 53 s.

[6] Koivisto, J. Microsoft Excelin laajentaminen ohjelmistojen mallinnustyökaluksi. Julkaisematon. Tampereen teknillinen yliopisto. Diplomityö.

[7] Koivula, J. Microsoft Wordin laajentaminen asiakirjojen käsittelyn kannalta. Tampere 2010. Tampereen teknillinen yliopisto. Kandidaatintyö. 23 s.

[8] Koskimies, K., Koskinen, J., Maunumaa, M., Peltonen, J., Selonen, P., Siikarla, M., Systä, T.: UML työvälineenä ja tutkimuskohteena. Tietojenkäsittelytiede. 21(2004), pp. 19–51

[9] Koskimies, K., Mikkonen, T. Ohjelmistoarkkitehtuurit. Helsinki 2005, Talentum Media Oy. 250 s.

[10] Microsoft. Word Object Model Overview. [WWW]. [Viitattu 5.5.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/kw65a0we.aspx

[11] Microsoft. Host Items and Host Controls Overview. [WWW]. [Viitattu 5.5.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/9z4e3456.aspx

[12] Microsoft. Office Solutions Development Overview. [WWW]. [Viitattu 5.5.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/hy7c6z9k.aspx

[13] Microsoft. Walkthrough: Calling Code in an Application-Level Add-in from VBA. [WWW]. [Viitattu 5.5.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/bb608614.asp

[14] Muhametsin, S. A Language-Independent Agent Architecture. Tampere 2011. Tampereen teknillinen yliopisto. Diplomityö. 49 s.

[15] Object Management Group. OMG Unified Modeling Language™ (OMG UML), Infrastructure. [WWW]. [Viitattu 5.5.2012]. Saatavissa: http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/

[16] Object Management Group. OMG Unified Modeling Language™ (OMG UML), Superstructure. [WWW]. [Viitattu 5.5.2012]. Saatavissa: http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF

[17] Purdy, D., Richter, J. Exploring the Observer Design Pattern. [WWW]. [Viitattu 5.5.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/ee817669.aspx

[18] Sparx Systems Pty Ltd. Enterprise Architect - UML Design Tools and UML CASE tools for software development. [WWW]. [Viitattu 7.5.2012]. Saatavissa: http://www.sparxsystems.com/products/ea/index.html

[19] Sybase Inc. PowerDesigner Blueprint Newsletter. [WWW]. [Viitattu 16.5.2012]. Saatavissa: http://www.sybase.com/content/1031403/PD_Blueprint_v13.pdf

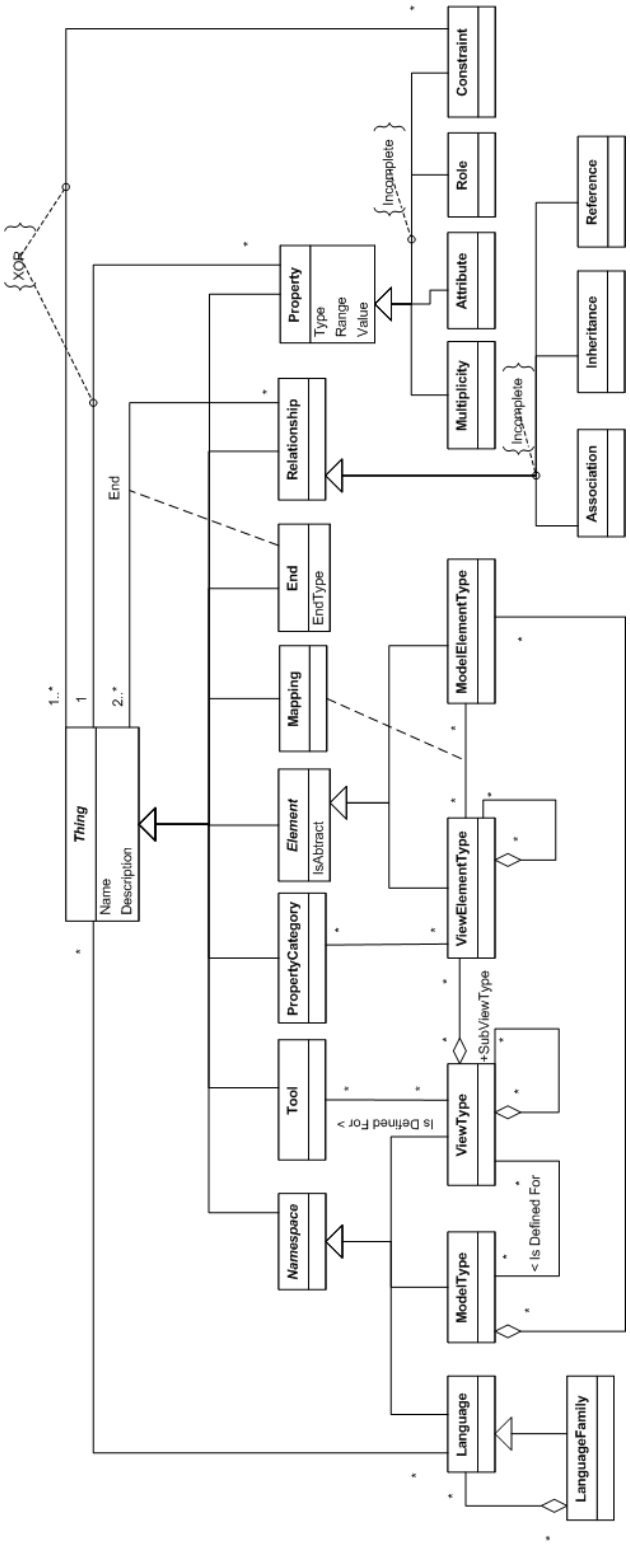
[20] Sybase Inc. PowerDesigner Modeling & Metadata Management Software Solution for all Architecture - Sybase Inc. [WWW]. [Viitattu 7.5.2012]. Saatavissa: http://www.sybase.com/products/modelingdevelopment/powerdesigner

[21] Sybase Inc. SyBooks Online. [WWW]. [Viitattu 7.5.2012]. Saatavissa: http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc00121.1600/doc/html/rad1232637364902.html

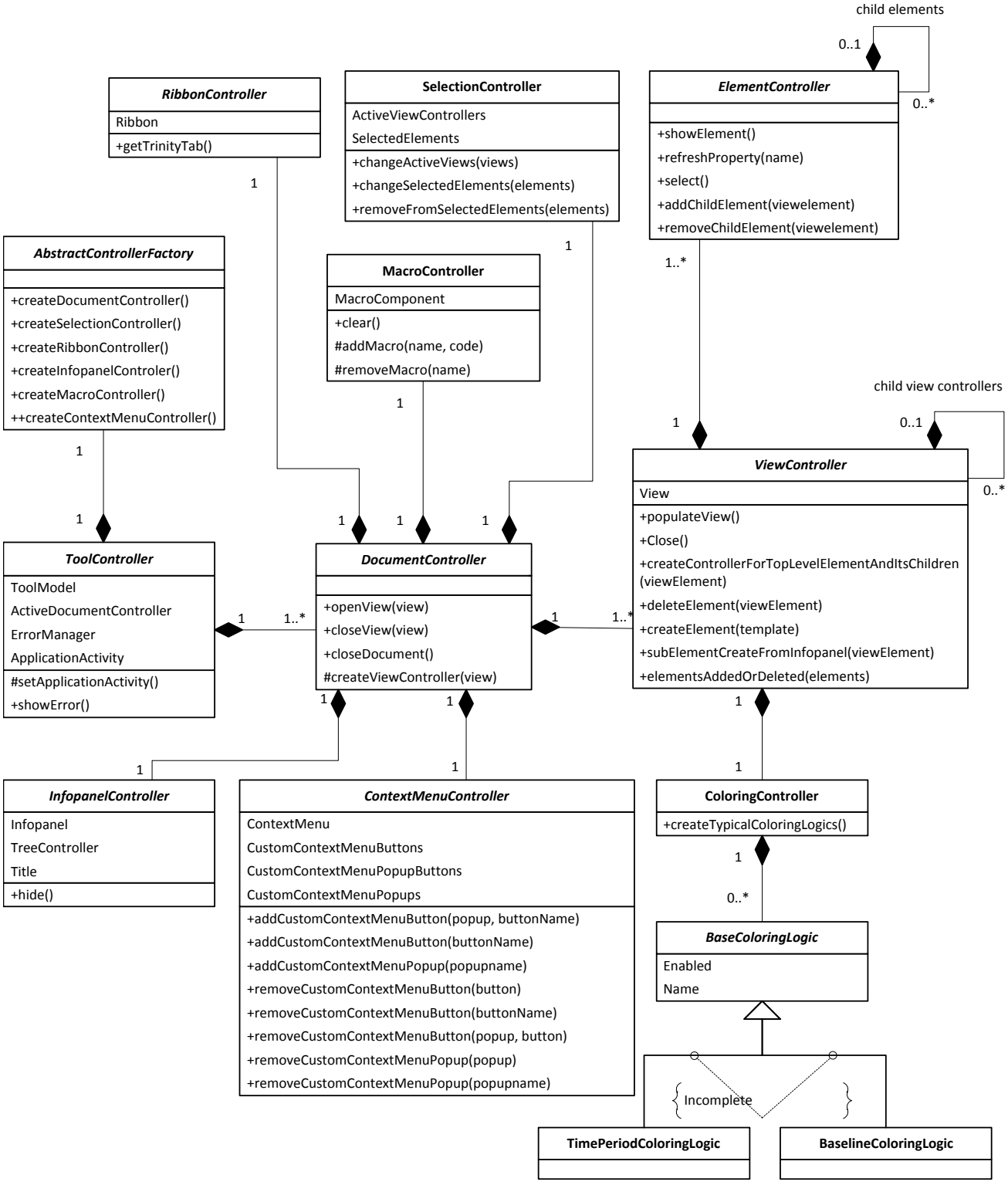
[22] Wollin, L. Overriding the Built-In Menus and Commands in Microsoft Word. [WWW]. [Viitattu 5.5.2012]. Saatavissa: http://msdn.microsoft.com/en-us/library/aa140285(office.10).aspx

[23] Xia, S., Sun, D., Sun, C., Chen, D., Shen, H.: Leveraging Single-user Applications for Multi-user Collaboration: the CoWord Approach. Word Journal Of The International Linguistic Association. (2004), pp. 162–171 2004

LIITE 1: TRINITYN METAMETAMALLI



LIITE 2: TOOL-OHJELMISTOKEHYKSEN VIEWCONTROLLER-MODUULIN ARKKITEHTUURI



LIITE 3: WORD-LAAJENNOKSELLA LUOTU ESIMERKKIKÄYTTÖTAPAUS

Example use case

Use case: Opening a view, simple case

Description: The user wants to open an existing view in its default tool

Actor: The user

Purpose:

Status:

Version: 0.5

Requirement type:

Step 1: Expand MUI

Description: The user expands the MUI from the upper left corner of the primary display

Transitions:

- MUI is expanded *( )*

Step 2: Double-click the view

Description: The user double-clicks the view in MUI's model viewer

Transitions:

- The default tool is opened *(Default tool not yet open)*

- The view is opened in the tool *(Default tool already open)*

Final state:

Description: The opened view is presented in the default tool.